PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-263447 = EP0718761
(43)Date of publication of application : 11.10.1996

(51)Int.Cl. G06F 15/16
G06F 9/445
G06F 13/00

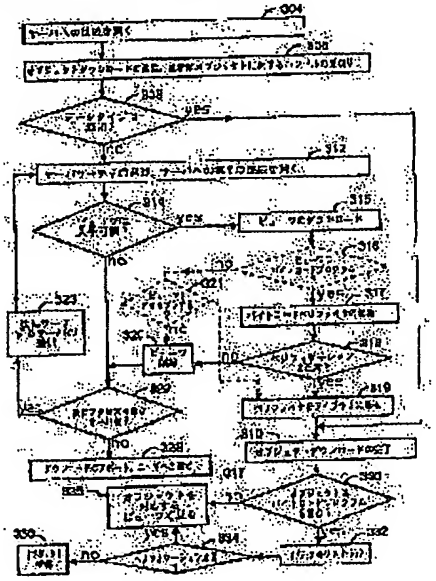
(21)Application number : 07-349164
(22)Date of filing : 20.12.1995

(71)Applicant : SUN MICROSYST INC
(72)Inventor : GOSLING JAMES A

(30)Priority
Priority number : 94 359884 Priority date : 20.12.1994 Priority country : US

(54) DISTRIBUTED COMPUTER SYSTEM AND OPERATION METHOD THEREFOR

(57)Abstract:
PROBLEM TO BE SOLVED: To provide a distributed computer system provided with a computer for automatically clown-loading the viewer of an object to be referred to and verifying the maintainability of a loaded program and the operation method.
SOLUTION: At the time of loading data (object) stored in another server or the like through a network and referring to them, the viewer corresponding to the object is automatically searched, and in the case that the appropriate viewer is found in the other server or the like, verification is performed so as to confirm the maintainability before activating the viewer. Especially, importance is placed in the verification relating to the use of a stack and a data type to the program written in a byte code language.



PF030016
corres to
EP0718761

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平8－263447

(43)公開日　平成８年(1996)10月11日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	15/16	３７０		Ｇ０６Ｆ 15/16	３７０Ｎ	
	9/445		7368－5Ｅ	13/00	３５１Ｈ	
	13/00	３５１		9/06	４２０Ｊ	

審査請求　未請求　請求項の数４　ＦＤ　（全 34 頁）

(21)出願番号　特願平7－349164

(22)出願日　平成７年(1995)12月20日

(31)優先権主張番号　０８／３５９，８８４

(32)優先日　1994年12月20日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　595034134
サン・マイクロシステムズ・インコーポレイテッド
Ｓｕｎ Ｍｉｃｒｏｓｙｓｔｅｍｓ，Ｉｎｃ．
アメリカ合衆国カリフォルニア州94043－1100・マウンテンビュー・ガルシアアベニュー　2550

(72)発明者　ジェイムス・エイ・ゴスリング
アメリカ合衆国カリフォルニア州94062・ウッドサイド・ピーオーボックス　620509

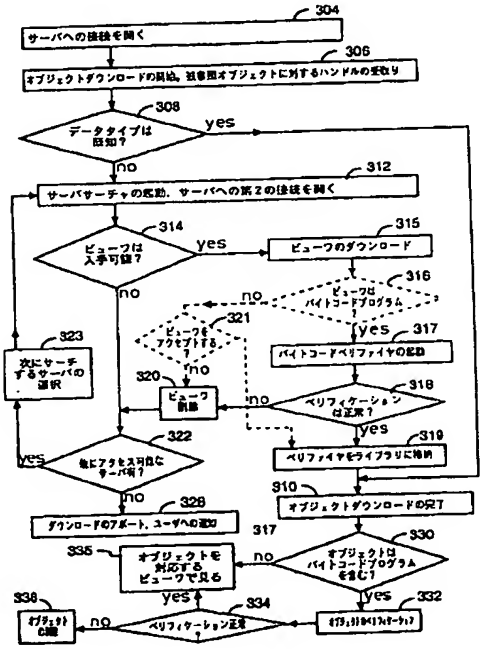
(74)代理人　弁理士　大島　陽一　（外１名）

(54)【発明の名称】　分散型コンピュータシステム及びその動作方法

(57)【要約】

【課題】　参照しようとするオブジェクトのビューワを自動的にダウンロードするとともにロードしたプログラムの保全性をベリファイするコンピュータを含む分散型コンピュータシステム及びその動作方法を提供すること。

【解決手段】　本発明は、他のサーバなどに格納されているデータ（オブジェクト）をネットワークを介してロードして参照するとき、そのオブジェクトに対応するビューワを自動的にサーチし、他のサーバなどで適切なビューワが見つかった場合、そのビューワを起動する前に、保全性を確認するためベリフィケーションを行うことを主な特徴とする。本発明は、特にバイトコード言語で書かれたプログラムに対するデータタイプとスタックの使用に関するベリフィケーションに重点を置いている。



1

【特許請求の範囲】

【請求項１】　複数の個別のコンピュータを有する分散型コンピュータシステムの動作方法であって、

（ａ）第１コンピュータに、各々関連するデータタイプのオブジェクトを見るための複数のビューワプログラムを格納する過程と、

（ｂ）前記第１コンピュータに於いて、第２コンピュータ内に格納されているオブジェクトへのリファレンスを選択する過程と、

（ｃ）前記第１コンピュータに於いて、前記リファレンスの選択に応じて、前記第１コンピュータと前記第２コンピュータとの間に第１接続リンクを確立し、前記オブジェクトに関連するデータタイプ情報の検索を含む、前記第２コンピュータからの前記オブジェクトの検索を開始する過程と、

（ｄ）前記第１コンピュータに於いて、前記第１コンピュータに格納された前記ビューワプログラムの中に、前記検索されたデータタイプに対応するビューワプログラムが含まれているかどうか判定する過程と、

（ｅ）前記過程（ｄ）での判定結果が否定の場合、前記検索されたデータタイプに対応するビューワプログラムが、前記第１コンピュータを含む他のコンピュータのセット内に格納されているかどうか判定する過程と、

（ｆ）前記過程（ｅ）での判定結果が肯定の場合、（ｆ１）前記第２データタイプに対応する前記ビューワプログラムのコピーを前記第１コンピュータにロードする過程と、（ｆ２）前記コピーされたビューワプログラムに対しベリフィケーションプロシージャを実行して、前記コピーされたビューワプログラムが予め定められたオベランドスタック使用基準に合うかどうか判定する過程と、（ｆ３）前記過程（ｆ２）での判定結果が肯定の場合、前記第２オブジェクトを見ることができるように前記コピーされたビューワプログラムを実行する過程とを含むことを特徴とする方法。

【請求項２】　前記過程（ｅ）が、

前記検索されたデータタイプに対応するビューワプログラムが前記第２コンピュータに格納されているかどうか最初に判定し、前記判定の結果が否定の場合、前記第２データタイプに対応するビューワプログラムが予め定められた他のコンピュータのセットの何れかの中に格納されているかどうか判定し、前記第２の判定結果が肯定の場合、前記過程（ｆ１）乃至（ｆ３）を実行する過程を含んでいることを特徴とする請求項１に記載の方法。

【請求項３】　複数の個別のコンピュータを有する分散型コンピュータシステムであって、

第１コンピュータと第２コンピュータを含み、

前記第１コンピュータが、オブジェクト及びビューワプログラム格納用第１メモリを有し、格納されている前記オブジェクトの各々は関連するデータタイプ情報を含み、

2

前記第２コンピュータが、

ビューワプログラム格納用第２メモリと、

前記第１コンピュータの前記第１メモリ内に格納されている前記オブジェクトの一つへのリファレンスを、前記第２コンピュータのユーザが選択することができるようにするためのユーザインタフェース制御プログラムと、

前記オブジェクトリファレンスの選択に応じて、前記第２コンピュータと前記第１コンピュータとの間に第１接続リンクを確立し、前記オブジェクトに関連するデータタイプ情報の検索を含む、前記第１コンピュータからの前記オブジェクトの検索を開始するコンピュータ間リンク制御プログラムとを含み、

前記第１及び第２コンピュータに含まれる各ビューワプログラムは、対応するデータタイプのオブジェクトを見ることを可能とし、

前記ユーザインタフェース制御プログラムが、前記第２コンピュータに含まれている前記ビューワプログラムに、前記検索されたデータタイプに対応するビューワプログラムが含まれているかどうか判定して、判定結果が否定の場合、前記検索されたデータタイプに対応するビューワプログラムを前記第１コンピュータを含む他のコンピュータのセットの中から見つけようと試みるビューワサーチ命令を含み、

前記コンピュータ間リンク制御プログラムが、前記ビューワサーチ命令が前記検索されたデータタイプに対応するビューワプログラムを前記他のコンピュータ内に見つけたとき、そのビューワプログラムのコピーを前記第２コンピュータ内にロードするためのビューワダウンロード命令を含んでいることを特徴とするシステム。

【請求項４】　前記第２コンピュータが、更に、

前記コピーされたビューワプログラムが、予め定められたオベランドスタック使用基準に合うかどうか判定するためのベリフィケーションプロシージャと、

前記ベリフィケーションプロシージャが前記コピーされたビューワプログラムが前記予め定められた基準に合うと判定したとき、前記第２オブジェクトを見るための前記コピーされたビューワプログラムの実行をイネーブルするプログラムイネーブル命令とを含むことを特徴とする請求項３に記載のシステム。

【発明の詳細な説明】

【０００１】

【発明が属する技術分野】本発明は、一般に、個別の基本マシン命令セットを用いる複数のコンピュータプラットフォーム上のコンピュータソフトウェアの利用法に関する。特に、ネットワークサーバまたは他のソースから得られるコンピュータソフトウェアの保全性をベリファイ（検証）する方法に関する。

【０００２】

【従来の技術】図１に一般的に示されているように、典型的な従来のネットワーク化されたコンピュータシステ

ム１００に於いて、第１コンピュータ１０２は、第２コンピュータ１０４にあるコンピュータプログラム１０３をダウンロードすることができる。この例では、第１ユーザノード１０２は、典型的には、中央演算装置（ＣＰＵ）１０６、ユーザインタフェース１０８、プログラム実行用の主メモリ１１０（例えばＲＡＭ）、オペレーティングシステム１１３、文献及び他のデータを格納するための補助記憶装置１１２（例えばハードディスク）、及びコンピュータネットワーク１２０（例えばインターネット、ローカルエリアネットワークまたはワイドエリアネットワーク）に接続するためのモデムなどのコミニュケーションインタフェース１１４を有するユーザワークステーションである。コンピュータ１０２及び１０４は、“ネットワーク上のノード”または“ネットワークノード”と呼ばれることもある。

【０００３】第２コンピュータ１０４はネットワークサーバであるのが一般的だが、第２ユーザワークステーションであっても良く、通常、第１コンピュータと基本的に同じコンピュータ要素を含んでいる。

【０００４】従来、第１コンピュータ１０２が第２コンピュータ１０４からコンピュータプログラム１０３のコピーをダウンロードした後に、第１コンピュータ１０２のユーザが、ダウンロードされたプログラム１０３の保全性をベリファイするのを支援するための標準的なツールは実質的にない。特に、第１コンピュータのユーザが、ダウンロードされたプログラムのソースコードを調べない限り、従来技術のツールを用いて、ダウンロードされたプログラム１０３がスタックに対しアンダーフローやオーバフローとならないか、或いはダウンロードされたプログラム１０３がユーザコンピュータのファイルや他のリソースに悪影響を与えないかどうかを判断することは実質的に不可能である。

【０００５】あるコンピュータから別のコンピュータへのコンピュータソフトウェアのダウンロードに関する別の問題は、個別の基本マシン命令セットを用いているコンピュータプラットフォーム間のコンピュータソフトウェアの転送の問題である。従来も、プラットフォームに依存しないコンピュータプログラムやプラットフォームに依存しないコンピュータプログラミング言語の例はいくつかある。従来技術に欠けているのは、ネットワークサーバまたは他のソースから得られる、プラットフォームに依存しないコンピュータソフトウェアの保全性を、そのソフトウェアを受け取った人がベリファイするのを可能にする信頼の置ける自動ソフトウェアベリフィケーション（verification）ツールである。

【０００６】本発明の別の側面は、遠隔からダウンロードするオブジェクトまたはファイルをユーザが選択した後に、そのオブジェクトまたはファイルに関連するソフトウェアを自動的にダウンロードするための方法に関する。例えば、“ワールドワイドウェブ（World Wide Web：WWW）”として知られている、インターネットで広く用いられている機能があるが、このWWW上で、ある文献を検索しているとき、その文献のあるページに、他の文献またはオブジェクトへのリファレンス（参照）が含まれていることがある。ユーザは、所与のオブジェクトを選択することによって、関連するハイパーリンクを介し、それらの文献またはオブジェクトにアクセスすることができる。そのような選択は、通常、ユーザがワークステーションノードのグラフィカルユーザインタフェースを使い、ポインタデバイスを用いてハイパーリンクを表すグラフィックイメージを指定し、ポインタデバイス上のボタンを押すことによって行われる。ハイパーリンクの選択に応答して、ユーザのウェブアクセスプログラムは、参照したい文献またはオブジェクトが格納されているサーバ（その時見ている文献またはオブジェクト内のハイパーリンクに組み込まれているデータによって示されている）への接続を開き、それらの文献またはオブジェクトをダウンロードする。しかしながら、ダウンロードされる文献またはオブジェクトのデータタイプが、ユーザのウェブアクセスプログラムにとって既知でない場合、ユーザはダウンロードした文献を見ることができないか、或いは見ることはできても用いることができない。

【０００７】このようなことが起きると、ユーザは、しばしば文献またはオブジェクトを検索してきたサーバや、或いは他のサーバのプログラムライブラリを調べて、ダウンロードした文献またはオブジェクトに対応するビューワ（viewer）をマニュアルで見つけようと試みるかもしれない。ユーザのコンピュータプラットフォームとコンパチビリティ（compatibility）のあるビューワが見つかると、ユーザはそのビューワをダウンロードし、それを起動（実行）して、前にダウンロードしたオブジェクトを見ようとするだろう。しかしながら、出所のはっきりしないビューワを実行することには、かなり大きなリスクがある。例えば、ダウンロードしたビューワプログラムが“ウィルス”に感染しており、ユーザのコンピュータの保全性を損なうかもしれない。或いは、ユーザの期待に反して、ダウンロードしたプログラム自身がリソースへのアクセス及び／またはユーザコンピュータのデータの破壊を行う可能性もある。本発明は、文献乃至オブジェクトに対応するビューワを自動的にダウンロードするとともに、ダウンロードしたビューワを起動する前に、そのプログラムの保全性を自動的にベリファイすることによってこのような問題を解消するものである。

【０００８】

【発明が解決しようとする課題】上述したように、本発明の主な目的は、参照しようとする、他のコンピュータ（サーバなど）に含まれる文献またはオブジェクトに対応するビューワを自動的にダウンロードするとともに、

ダウンロードしたビューワを起動する前に、そのプログラムの保全性を自動的にベリファイするコンピュータを含む分散型コンピュータシステム及びその動作方法を提供することである。

【０００９】

【課題を解決するための手段】本発明は、遠隔のコンピュータノードからオブジェクト及びオブジェクトビューワを検索（即ちダウンロード）したり、オブジェクトを見るべく、格納したオブジェクトビューワを起動したりするための“クラスローダ（class loader）”に関する。ユーザが、例えばWWWのハイパーリンク機能を用いることによって見たいオブジェクトを選択すると、従来と同様に、参照されるオブジェクト（被参照オブジェクトとよぶこともある）のダウンロードが開始される。しかしながら、本発明のクラスローダは、オブジェクトのダウンロードプロセス開始時に受け取るデータタイプ情報を用いて、参照するオブジェクトに対応するビューワが、ユーザワークステーションに於いて入手可能かどうかをまず調べる。

【００１０】適切なビューワがそのワークステーションに於いて得られない場合、クラスローダは、オブジェクトをダウンロードしているサーバ、或いはユーザのワークステーションが認識している他の適切なサーバから適切なビューワを自動的に見つける。クラスローダは、見つけたビューワをダウンロードした後、このビューワを起動する前に、その保全性をベリファイするためプログラムベリフィケーションプロシージャを呼び出す。いったんベリファイされると、ビューワはユーザのローカルビューワライブラリに加えられ、被参照オブジェクトのダウンロードは終了し、ダウンロードしたオブジェクトを見るためのビューワの起動がイネーブルされる。

【００１１】適切なビューワが見つからないか、或いは見つけた唯一のビューワがベリフィケーションプロシージャを通過できない場合は、参照しようとしていたオブジェクトのダウンロードはアボートされる。

【００１２】本発明では、限定された、データタイプに依存したバイトコードセットを用いたバイトコード言語（“ＯＡＫ言語（ＯＡＫ language）”として商品化されている）で書かれたコンピュータプログラムの保全性をベリファイする。この言語で使用可能なソースコードのバイトコードは、（Ａ）処理可能なデータタイプについて制約があるスタックデータ消費型バイトコードである、（Ｂ）スタックデータは使用しないが、データタイプが既知であるデータをスタックに加えたり、データタイプに関係なくスタックからデータを取り出したりすることによってスタックに影響を与える、或いは（Ｃ）スタックデータの使用も、スタックへのデータの付加もしないのいずれかである。

【００１３】本発明によると、バイトコードプログラムを実行する前に、データタイプがマッチしていないデータを処理しようとする命令がないかどうか、あるいはその指定されたプログラム内のバイトコード命令の実行によってオペランドスタックのアンダーフローやオーバフローが生じないかどうかをチェックし、そのようなプログラムを使用するのを避けるための方法及びベリファイヤが提供される。

【００１４】本発明のバイトコードプログラムベリファイヤは、指定されたバイトコードプログラムを実行するとき、プログラムオペランドスタック内に格納されるデータを示すスタック情報を一時的に格納するための仮想スタック（virtual stack）を含んでいる。このベリファイヤは、プログラムの各ポイントに於いてオペランドスタックに格納されるであろうデータの数、順番、及びデータタイプを示すように仮想オペランドスタックを更新しながらプログラムの各バイトコード命令を順次処理していくことによって、指定されたプログラムを処理する。このベリファイヤは、仮想スタックの情報と、各バイトコード命令のデータタイプに関する制約とを比べて、プログラムを実行したときオペランドスタックにバイトコード命令のデータタイプに関する制約に適合しないデータが含まれるようなことがないかどうか判定するとともに、指定されたプログラム内のバイトコード命令によってオペランドスタックのアンダーフローやオーバフローが生じるようなことはないかどうかチェックする。

【００１５】バイトコードプログラムの命令の流れを細かく分析しなくていいように、またバイトコード命令を何度もベリファイしなくていいように、指定されたプログラム内に於いて別個の２以上のバイトコードの実行の直後に実行され得る全てのポイント（多重エントリポイントと呼ぶ）がリストアップされる。一般に、そのような先行する２以上の別個のバイトコードの少なくとも一つは、ジャンプ／分岐バイトコードである。指定されたプログラムの処理時、ベリファイヤは、各多重エントリポイントの直前（即ち、先行するバイトコード命令の一つの後）の仮想オペランドスタックの“スナップショット”をとり、このスナップショットを、同じ多重エントリポイントに対する他の先行するバイトコード命令の各々を処理した後の仮想スタックステータスと比較して、仮想スタックステータスが同一でない場合はプログラムフォールト（program fault）を生成する。

【００１６】

【発明の実施の形態】本発明の好適実施例について詳細に説明する。これらの例は添付の図面に図示されている。本発明を好適実施例に基づいて説明するが、理解されるように、それらは本発明をそれらの実施例に限定することを意図するものではない。本発明は、特許請求の範囲によって画定される本発明の精神及び範囲に含まれるべき変形変更及び等価物を含むものである。

【００１７】図２に示す分散型コンピュータシステム２

００を参照されたい。第１コンピュータノード２０２は、例えばインターネットのようなコンピュータコミュニケーションネットワーク２１６を介して第２コンピュータノード２０４とつながっている。第１コンピュータノード２０２は、中央演算装置（ＣＰＵ）２０６、ユーザインタフェース２０８、主メモリ（ＲＡＭ）２１０、補助記憶装置（ディスク記憶装置）２１２、及び第１コンピュータノード２０２をコンピュータコミュニケーションネットワーク２１６につなげるためのモデムその他のコミュニケーションインタフェース２１４を含んでいる。ディスク記憶装置２１２は、データファイルその他の情報と共に、ＣＰＵ２０６によって実行されるプログラムを格納する。

【００１８】第２コンピュータノード２０４（ここでは、ファイルや他の情報のサーバとして構成されているものとする）は、ＣＰＵ２１８、ユーザインタフェース２２０、主メモリ（ＲＡＭ）２２２、補助記憶装置（ディスク記憶装置）２２４、及びコンピュータコミュニケーションネットワーク２１６と第２コンピュータノードとを接続するためのモデムその他のコミュニケーションインタフェース２２６を含んでいる。ディスク記憶装置２２４は、ファイル及び／またはオブジェクトディレクトリ２２８（ディスクディレクトリまたはカタログと呼ばれることもある）を含んでいる。ディスクディレクトリ２２８は、補助記憶装置２２４内に格納されている情報、即ち、ＣＰＵ２１８によって実行されたり他のコンピュータノードに配布されたりするプログラム２３４、ビューワライブラリ２３２、及びオブジェクト２３０の位置を示す。

【００１９】第１コンピュータノード２０２及び第２コンピュータノード２０４は、それぞれが異なるコンピュータプラットフォーム及びオペレーティングシステム２３６、２３７を用い、これらの２つのコンピュータノードの一方で実行されるオブジェクトコードプログラムが、他方のコンピュータでは実行できないことがあってもよい。例えば、サーバノード２０４をＵｎｉｘオペレーティングシステムを用いたサン・マイクロシステムズ社のコンピュータとし、ユーザワークステーションノード２０２を８０４８６マイクロプロセッサとマイクロソフト社のＤＯＳオペレーティングシステムを用いたＩＢＭ互換機とすることもできる。更に、同じネットワークに接続され、同じサーバ２０４を利用する他のユーザワークステーションに於いて、様々なコンピュータプラットフォーム及びオペレーティングシステムを用いてもよい。

【００２０】これまで、様々なタイプのコンピュータを有するネットワークでソフトウェアを配布するためのサーバ２０４は、各コンピュータプラットフォーム（例えば、Ｕｎｉｘ、ウィンドウズ、ＤＯＳ、マッキントッシュなど）に対して個別のソフトウェアライブラリ（例えば個別のビューワライブラリ２３２）を有していた。その結果、様々なシステムのユーザの要望に応えるため、サーバは同じコンピュータプログラムの様々なバージョン（例えば２３８、２３９）と、複数のオブジェクトビューワ（例えば２４１、２４３）の両方を、各コンピュータプラットフォームタイプに対し一つずつ格納する必要があった。しかしながら、本発明を用いると、一種類のバイトコードバージョンのプログラムを配布することによって、多様なユーザをサポートすることができる。

【００２１】図３に、本発明に従った分散型コンピュータシステム２５０を示す。第１コンピュータノード２５２は、例えばインターネットのようなコンピュータコミュニケーションネットワーク２６６を介して第２コンピュータノード２５４につながっている。ここでもまた、従来と同様に、第１コンピュータノード２５２と第２コンピュータノード２５４が、別個のコンピュータプラットフォーム及びオペレーティングシステム２５５、２５６を用い、一方のコンピュータで実行されるオブジェクトコードプログラムが他方のコンピュータで実行できなくても良い。例えば、図２を参照して上述したように、サーバノード２５４がＵｎｉｘオペレーティングシステムを用いたサンマイクロシステムズ社のコンピュータで、ユーザワークステーションノード２５２が８０４８６マイクロプロセッサとマイクロソフト社のＤＯＳオペレーティングシステムを用いたＩＢＭ互換機であってもよい。第１コンピュータノード２５２は、ＣＰＵ２５７、ユーザインタフェース２５８、主メモリ（ＲＡＭ）２６０、補助記憶装置（ディスク記憶装置）２６２、及びコンピュータコミュニケーションネットワーク２６６に第１コンピュータノード２５２を接続するためのモデムその他のコミュニケーションインタフェース２６４を含んでいる。ディスク記憶装置２６２は、ＣＰＵ２５７によって実行されるプログラムを格納する。これらのプログラムの少なくとも一つは実行可能形式のバイトコードプログラム２６７である。説明のため、第１コンピュータノード２５２が、第２コンピュータノード２５４からコンピュータコミュニケーションネットワーク２６６を介してバイトコードプログラム２６７を受け取ることとする。その詳細について、クラスローダと共に以下に説明する。

【００２２】好適実施例では、バイトコードプログラムはＯＡＫアプリケーションとして書かれており、コンパイルまたはインタープリットされて、一連の実行可能な命令になる。ＯＡＫ命令セットの全ソースコードバイトコード命令（source code bytecode instructions）のリストを表１に示す。ＯＡＫ命令セットは、データタイプに依存する（即ちデータタイプが特定されている）バイトコード命令によって特徴付けられる。詳述すると、ＯＡＫ命令セットでは、異なるデータタイプに対する同じ基本操作に別個の操作コード（opcode）を付すことに

よって、それらを区別している。その結果、同じ基本機能（例えば２つの数字の加算）を実行するために、それぞれ対応する特定のデータタイプのデータだけを処理する複数のバイトコードが命令セット内に含まれている。更に、ＯＡＫ命令セットに含まれていない命令についても注意されたい。例えば、ＯＡＫ言語命令セットには“計算されたｇｏｔｏ（computed goto）”命令はなく、オブジェクトリファレンスを変更したり、新たなオブジェクトリファレンスを生成する命令もない（既存のオブジェクトリファレンスのコピー命令は除く）。ＯＡＫ命令セットに関するこれらの２つの制約は、他の制約とともに、ＯＡＫ命令セットに含まれるデータタイプが特定された命令と同じようにデータを用いるバイトコードプログラムであれば、ユーザのコンピュータシステムの保全性を損なうことはないということを保証するのに役立っている。

【００２３】好適実施例では、使用可能なデータタイプは、整数、倍長整数（long integer）、ショートインテジャ（１６ビット信号の整数）、単精度浮動小数点数、倍精度浮動小数点数、バイト、キャラクタ、及びオブジェクトポインタ（本明細書中では、オブジェクトリファレンスと呼ぶこともある）がある。各“オブジェクトリファレンス”データタイプはデータタイプの一部としてオブジェクトクラス仕様（object class specification）を含むことができるため、“オブジェクトリファレンス”データタイプに含まれるデータのサブタイプの数には実質的に制限がない。更に、プログラム内で用いられる定数もデータタイプを有しており、好適実施例で使用される定数のデータタイプには、上述したデータタイプに加えて、class、fieldref、methodref、string、及びAscizが含まれる。それらは全て特定の目的を持った２バイト以上の定数データを表す。

【００２４】データタイプに依存しないバイトコードの中には、スタック操作機能を有するものがいくつかある。スタック操作機能には（Ａ）スタック上の１または複数のワードを複製して、それらをスタック内の特定の位置に配置し、それによってデータタイプが既知のより多くのスタックアイテム（stack item）を生成する機能、（Ｂ）１または複数のアイテムをスタックから消去する機能などが含まれる。他の少数のデータタイプに依存しないバイトコードには、スタック上のワードを全く用いずスタックを変化させなかったり、あるいは、スタック上にあるワードを使用することはないがスタックにワードを加えたりするものがある。これらのバイトコードは、それらが実効される前のスタックの内容についてデータタイプに関する制約はないが、スタック内のアイテムのデータタイプに関してスタックの内容を変化させる。ただし、その変化のさせかたは全く予測可能である。従って、スタック内の全てのオペランドのデータタイプ及びその数は、１００％の精度で常に予測（即ち計算）可能である。

【００２５】第２コンピュータノード２５４（ここではファイルまたは他の情報サーバとして構成されているものとする）は、ＣＰＵ２６８、ユーザインタフェース２７０、主メモリ（ＲＡＭ）２７２、補助記憶装置（ディスク記憶装置）２７４、及びコンピュータコミュニケーションネットワーク２６６に第２コンピュータノードを接続するためのモデムその他のコミュニケーションインタフェース２７６を含んでいる。ディスク記憶装置２７４は、ディスクディレクトリ２８０、第１オブジェクト２８３を含むオブジェクト２８２、ビューワライブラリ２８４、及びＣＰＵ２６８によって実行されたり、及び／または他のコンピュータノードに配布されたりするプログラム２８６（それらのうちの少なくとも一つはコンピュータノード２５２に転送されるバイトコードプログラム２６７である）を含んでいる。

【００２６】図３に示されているように、第１コンピュータノード２５２の補助記憶装置２６２内には、他のコンピュータノードからオブジェクトビューワやオブジェクトを検索（即ちダウンロード）し、格納したオブジェクトビューワを呼び出してオブジェクトを見るためのクラスローダプログラム２９６が格納されている。また、クラスローダ２９６は、ダウンロードされたオブジェクトビューワがユーザによって起動される前に、そのビューワを（エンドユーザワークステーションノードに於いて）自動的にベリファイし、その保全性をチェックする。

【００２７】この特許出願の目的に於いては、対応するビューワを用いることによって“見る”ことが可能な“オブジェクト”は、（Ａ）ＪＰＥＧ、ＧＩＦ、ＭＰＥＧ、あるいはＭＰＥＧ２データといった特定のタイプまたはフォーマットのデータを含むファイルその他のデータ構造の、メソッド（method）またはソフトウェアが組み込まれていないデータオンリー型オブジェクト（data-only type of object）、あるいは（Ｂ）１または複数のメソッドが組み込まれており、オプションとしてデータも含み得るファイルその他のデータ構造の、メソッド格納型オブジェクト（method-storing object）のいずれかである。例えばＪＰＥＧやＧＩＦのようなイメージデータタイプを格納しているデータオンリー型オブジェクトを見るため、あるいは、例えばＭＰＥＧやＭＰＥＧ２のようなビデオプログラムデータタイプを格納しているデータオンリー型オブジェクトを見るためには、特定のビューワが必要であろう。他の例として、データの図表を見るためのビューワや、（解読のキーがユーザに既知の場合）暗号化されたデータを見るためのデータ解読ソフトウェアが含まれたビューワなどがあり得る。

【００２８】更に、様々なタイプの内部プログラムを用いたメソッド格納型オブジェクトに対して特定のビューワが必要となることがある。例えば、様々なメソッド格

納型オブジェクトの様々なタイプの内部プログラムが、特定の記述言語を用いていたり、ユーティリティプログラムの様々なライブラリの使用を仮定していたりすることによって、特定のビューワが必要となることがある。

【００２９】“ビューワ（インタープリタと呼ぶこともある）”は、指定されたオブジェクト内のデータ及び／または命令のデコードを行う。また、一般に特定のデータタイプまたはクラスのオブジェクトを使用可能にするために必要とされている計算及び処理を全て実行する。本発明では、そのようなオブジェクトビューワは、ソースコードバイトコード言語（source code bytecode language）で書かれたバイトコードプログラムであり、各オブジェクトビューワの保全性は、バイトコードプログラムベリファイヤ２４０を実行することによって、エンドユーザにより個別にベリファイすることができる。バイトコードプログラムのベリフィケーションについて、以下により詳細に説明する。

【００３０】分散型コンピュータシステム２５０は、本発明によるプラットフォームに依存しないオブジェクトビューワを含むことができるとともに、プラットフォームに依存し、本発明によるバイトコードプログラムベリファイヤ２４０及びクラスローダ２９６を用いてベリファイすることのできない他のオブジェクトビューワも含むことができるという点に注意されたい。このようなハイブリッドシステムでは、本発明による自動ビューワ保全性ベリフィケーションは、バイトコードビューワプログラムに対しては利点があるが、他のビューワプログラムに対しては利点がない。

【００３１】クラスローダ２９６は、遠隔にあるサーバからのオブジェクトビューワ及びオブジェクトのロード及びベリフィケーション用実行可能プログラムである。例えば、インターネットのWWW上である文献を検索するとき、その文献のあるページに他の文献またはオブジェクトへのリファレンスが含まれていることがある。ユーザは、そのような他の文献またはオブジェクトに、関連するハイパーリンクを介して所与のオブジェクトを選択することによってアクセスすることができる。そのような選択操作は、通常、ワークステーションノード上のグラフィカルユーザインタフェースを利用して、ポインタデバイスを用いてハイパーリンク選択を表すグラフィックイメージを指し示し、ポインタデバイス上のボタンを押すことによって、ユーザにより為される。

【００３２】選択プロセスに於いて、そのとき見ている文献またはオブジェクトにリファレンスが含まれている他の文献またはオブジェクトの中には、データタイプがユーザのワークステーションに既知でないものもあるだろう。本発明のクラスローダは“既知でない”データタイプに対応するビューワを見つけると共に、ダウンロードされたバイトコードプログラムがユーザによって実行される前に、それらの保全性をベリファイするのに用いられる。

【００３３】クラスローダ２９６には３つの主要な機能がある。第１に、クラスローダはダウンロードされたオブジェクト（及びそれらの関連するバイトコードプログラム）のデータタイプをチェックし、ユーザワークステーションが、その固有の記憶装置２６２に格納されている“ビューワライブラリ”２９８内に、対応するビューワを有しているかどうかを調べる。第２に、適切なビューワを見つけることができなかった場合、クラスローダは、ソースサーバと、認識している他のサーバとに対してサーチルーチンを実行し、適切なビューワを見つけてダウンロードすることを試みる。ビューワを見つけることができなかった場合は、ダウンロードされているオブジェクト及び／またはバイトコードプログラムを、適切なビューワがないという理由でリジェクトする。最後に、リモートソースに適切なビューワが見つかった場合、クラスローダは、バイトコードベリファイヤ２４０を起動し、バイトコードプログラムインタープリタ２４２によってビューワが実行される前に、またはバイトコードプログラムコンパイラ２４４によってコンパイルされる前に、ダウンロードされたビューワをチェックする。ベリフィケーション後、ダウンロードされたビューワは、ユーザのローカルビューワライブラリ２９８内に格納される。

【００３４】図３、図４及び付録１には、クラスローダプログラム２９６の動きが、関連するオブジェクトを介したバイトコードプログラムの検索に対し詳細に記載されている。付録１は、クラスローダプログラムを表す疑似コード（pseudocode）のリストである。付録１で用いられている疑似コードは、本質的には、従来の汎用コンピュータ言語を用いたコンピュータ言語と同等である。本願中で用いられている疑似コードは、本発明を説明する目的で作られたものであり、本分野で標準的な知識を有するコンピュータプログラマであれば容易に理解することができるように意図して書き下してある。

【００３５】図４に示されているように、ユーザワークステーション２５２は、ダウンロードされるオブジェクト２８３が格納されているサーバ２５４への接続を開き、ダウンロードプロセスを開始する（３０４）。クラスローダ２９６は、オブジェクトのハイパーリンク選択により、オブジェクトバイトコードプログラムの転送を開始し、サーバ２５４は参照されるオブジェクトに対する“ハンドル（handle）”を、ユーザワークステーション２５２に転送する（３０６）。ハンドルは、参照されるオブジェクト本体に先だって検索される。また、ハンドルには、オブジェクトのデータタイプ（オブジェクトクラスと呼ぶこともある）を含む、被参照オブジェクトの属性に関する情報が含まれる。

【００３６】最初のチェックに於いて、検索されるオブジェクトに関連するデータタイプが、ユーザシステムに

既知であるかどうかが判定される（３０８）。詳述すると、クラスローダは、ユーザワークステーション２５２の補助記憶装置２６２内にあるビューワライブラリ２９８をサーチして、特定のデータタイプを有するこのオブジェクトに対する適切なビューワが、アクセス可能かどうかチェックする。ビューワライブラリ２９８には、その時々に於いてユーザワークステーションによってアクセス可能な全てのデータタイプのビューワと、それらのメモリ内の位置とを示すリストが含まれている。このようにして、クラスローダは、参照されるオブジェクト本体を実際にダウンロードする前に、最初のハンドシェイクプロセスに於いてダウンロードされるオブジェクトを前処理し、そのオブジェクトとユーザワークステーションプラットフォームのコンパチビリティをチェックする。適切なビューワが見つかった場合、クラスローダは被参照オブジェクトのダウンロードを完了する（３１０）。

【００３７】適切なビューワが、ビューワライブラリ２９８内に見つからなかった場合（即ち、選択されたオブジェクトのデータタイプがユーザワークステーション２５２に既知でないことを意味する）、クラスローダは適切なビューワのサーチを行う。通常、ビューワのサーチを試みる最初の場所は、選択されたオブジェクトが格納されているのと同じサーバである。従って、クラスローダは、被参照オブジェクトが格納されているのと同じサーバに対して第２の接続を開き（３１２）、指示されているデータタイプに対応するビューワをリクエストする（３１４）。このサーバが適切なビューワを含んでいる場合、そのビューワはユーザワークステーションにダウンロードされる（３１５）。

【００３８】ビューワのダウンロードが完了すると、ダウンロードされたビューワがバイトコードプログラムの場合（３１６）、クラスローダは、バイトコードプログラムベリファイヤ２４０を呼び出して、そのビューワプログラムのベリフィケーションを開始する（３１７）。バイトコードプログラムベリファイヤ２４０は実行可能プログラムであり、ＣＰＵ２５７によってバイトコードプログラムが実行される前に、指定されたバイトコード（ソース）プログラムを、スタック操作の適正さや、オペランドデータタイプコンパチビリティについてベリファイする。バイトコードベリファイヤプログラム２４０の動作については、後により詳細に説明する。ベリフィケーション結果が正常の場合（３１８）、サーバサーチャ（server searcher）はベリファイされたオブジェクトビューワをビューワライブラリ２９８内に格納し、新たなデータタイプビューワが使用可能になったことが反映されるようにライブラリ内のディレクトリを更新する（３１９）。ベリフィケーションが正常でなかった場合、ダウンロードされたビューワは削除される（３２０）。

【００３９】本発明の実施例には、自動ダウンロードが可能で、かつベリファイ可能なオブジェクトビューワとベリファイ不能なオブジェクトビューワの両方を使用することができるものもある。このような実施例では、オブジェクトビューワのダウンロード（３１５）の後、ダウンロードされたオブジェクトビューワがバイトコードプログラムでない場合（３１６）、そのオブジェクトビューワをアクセプトするか否かに対し判定がなされる（３２１）。例えば、そのオブジェクトビューワをアクセプトするかどうかユーザに質問してもよく、あるいはそのようなオブジェクトビューワをアクセプトするかどうかを予め決めてコンフィギュレーションファイル（configuration file）に含ませておいてもよい。ベリファイ不能なオブジェクトビューワがアクセプトされる場合、そのオブジェクトビューワはビューワライブラリ内に格納される（３１９）。アクセプトされない場合、ダウンロードされたビューワは削除される（３２０）。

【００４０】サーバもユーザワークステーションも適切なビューワを格納しておらず、ステップ３０８及び３１４で、選択されたオブジェクトに対し用いるのに適切なビューワが見つからなかった場合、クラスローダは、ユーザワークステーションに認識されている他のサーバまたはリモートユーザワークステーション（例えば認識されているサーバのリスト３２７）を含むように探索範囲を広げる。再度図３を参照されたい。図３に示されている第２サーバ３２４は、ビューワライブラリ３２６を有する補助記憶装置（ディスク記憶装置）３２５を含んでいる。適切なビューワが第２サーバ３２４のビューワライブラリ３２６内で見つかった場合、クラスローダは、上述したステップ３１５～３２１に従って、そのビューワプログラムをダウンロードしベリファイする。クラスローダはこのプロセスを繰り返し、全ての認識されているリソースについて調べ尽くすか、または適切なビューワを見つけてベリファイするまで、順次サーバをチェックする。最終的に、適切なビューワを見つけることができなかった場合、被参照オブジェクトのダウンロードはアボートされ、被参照オブジェクトに対応するビューワが見つからなかったことをユーザに通知するべく、ユーザメッセージが生成される（３２８）。

【００４１】上述したように、適切なオブジェクトビューワが予めユーザワークステーション上のビューワライブラリ２９８に格納されていた場合（３０８）、或いは適切なオブジェクトビューワのダウンロード、ベリフィケーション、及びユーザビューワライブラリへの組み込みが成功した場合、選択されたオブジェクトのロードが完了する（３１０）。ダウンロードされたオブジェクトに１または複数のバイトコードプログラムが組み込まれている場合、従ってメソッド格納型オブジェクトである場合（３３０）、ダウンロードされたオブジェクト内のバイトコードプログラムは、これらの組み込みプログラ

ムに関し、バイトコードベリファイヤを実行することによりベリファイされる（３３２）。組み込みプログラムの処理が終了し、ベリファイヤが“成功”リターンコードを生成する場合（３３４）、ダウンロードされたオブジェクトは対応するオブジェクトビューワによって見ることができる（３３５）。ベリファイヤの要求に従わないプログラムを検出することによって、ベリファイヤが組み込みプログラムの処理をアボートした場合（３３４）、ダウンロードされたオブジェクトは削除され（３３６）、適切なユーザメッセージが生成される。

【００４２】ダウンロードされたオブジェクトに組み込みバイトコードプログラムが含まれていない場合（３３０）、ステップ３３２乃至３３４はスキップされ、オブジェクトは適切なビューワによって見ることができる（３３５）。

【００４３】再度図３を参照されたい。図示されているように、第１コンピュータノード２５２は、その補助記憶装置２６２内に、指定されたバイトコードプログラムの保全性をベリファイするためのバイトコードベリファイヤプログラム２４０と、指定されたバイトコードプログラムを実行するためのバイトコードインタープリタ２４２を格納している。別の手段として、または追加して、第１コンピュータノード２５２は、ベリファイされたバイトコードプログラムをオブジェクトコードプログラムに変換し、インタープリタ２４２より効率的にバイトコードプログラムを実効するバイトコードコンパイラ２４４を格納していてもよい。

【００４４】バイトコードベリファイヤ２４０は実行可能プログラムであり、バイトコードプログラムがバイトコードインタープリタ２４２の制御下でＣＰＵ２５７によって実行される前に（またはバイトコードプログラムがコンパイラ２４４によってコンパイルされる前に）、指定されたバイトコード（ソース）プログラムに於けるスタック操作の適正さや、オペランドデータタイプコンパチビリティをベリファイする。各バイトコードプログラム２６７（ダウンロードされたオブジェクトベリファイヤを含む）は、対応するベリフィケーションステータス３０２を有しており、この値はプログラムが他の場所からダウンロードされたとき、最初“偽（False）”にセットされる。プログラムに対するベリフィケーションステータス３０２は、プログラムがベリファイされ、ベリファイヤ２４０によって行われるデータタイプテスト及びスタックの使用に関するテストの全てに適合したときのみバイトコードベリファイヤ２４０によって“真（True）”にセットされる。

【００４５】バイトコードプログラムベリファイヤ

図５を参照されたい。バイトコードプログラムベリファイヤ２４０の処理について、特定のバイトコードプログラム３４０に関連して説明する。ベリファイヤ２４０は、ベリフィケーションプロセスに於いて必要とする情報を格納するため、いくつかの一時的なデータ構造（temporary data structure）を使用する。特に、ベリファイヤ２４０は、スタックカウンタ３４２、仮想スタック３４４、仮想ローカル変数アレイ３４５、及びスタックスナップショット格納構造３４６を使用する。

【００４６】スタックカウンタ３４２は、仮想スタック操作を追跡し、その時々の仮想スタック３４４のエントリ数が反映されるように、ベリファイヤ２４０によって更新される。

【００４７】仮想スタック３４４は、実際の実行時にバイトコードプログラム３４０によってオペランドスタック内に格納される各データに関するデータタイプ情報を格納する。好適実施例では、仮想スタック３４４は正規のスタックと同様に用いられるが、実際のデータ及び定数を記憶するのではなく、プログラムを実際に実行したときオペランドスタック内に格納される各データに対応するデータタイプを示す値を格納する点が異なる。従って、例えば、もし実際に実行したとき、スタックに３つの値：

HandleToObjectA

5

1

が格納されるとすると、対応する仮想スタックエントリは、

R

I

I

となる。ここで、仮想スタック内の“Ｒ”は、オブジェクトリファレンスを示し、仮想スタック内の各“Ｉ”は、整数を示す。更に、この例では、スタックカウンタ３４２は、仮想スタック３４４に格納されている３つの値に対応して、値３を格納する。

【００４８】各データタイプのデータには、対応する仮想スタックマーカ値（例えば、整数（Ｉ）、倍長整数（Ｌ）、単精度浮動小数点数（Ｆ）、倍精度浮動小数点数（Ｄ）、バイト（Ｂ）、ショート（Ｓ）、オブジェクトリファレンス（Ｒ））が割り当てられる。オブジェクトリファレンスに対するマーカ値には、しばしばオブジェクトクラス値が含まれる（例えば、Ｒ：ｐｏｉｎｔ、ここで“ｐｏｉｎｔ”はオブジェクトクラスである）。

【００４９】仮想ローカル変数アレイ３４５は、仮想スタック３４４と同じ基本機能を果たす。即ち、仮想ローカル変数アレイ３４５は、指定されたバイトコードプログラムが使用するローカル変数に対するデータタイプ情報を格納するのに用いられる。データは、しばしばプログラムによってローカル変数とオペランドスタックとの間で転送されるため、そのようなデータ転送を実行するか、あるいはそのようなデータ転送はしないがローカル変数を使用するようなバイトコード命令についてチェックすることにより、各バイトコード命令によってアクセ

スされるローカル変数が、これらのバイトコード命令に関するデータタイプの制約に適合するということを保証することができる。

【００５０】動作時、ベリファイヤ２４０は、スタックからのデータのポップ（pop）をリクエストする各データバイト命令を処理するとともに、同数のデータタイプ値を仮想スタック３４４からポップする。続いて、ベリファイヤは仮想スタック３４４から“ポップされた”データタイプ値と、バイトコード命令のデータタイプに関する要求とを比較する。同様に、スタックにデータをプッシュするようにリクエストする各バイトコード命令に対し、ベリファイヤは、対応するデータタイプ値を仮想スタックにプッシュする。

【００５１】本発明によるプログラムベリフィケーションの一側面によると、オペランドスタックステータス（operand stack status）に於けるオペランドの数とデータタイプが、ある特定の命令が実行されるとき常に同一となるかという点についてベリフィケーションがなされる。ある特定のバイトコード命令が２以上の異なる命令の直後に処理され得る場合、これらの異なる先行命令の各々を処理した直後の仮想スタックステータスを比較する必要がある。通常、これらの異なる先行命令の内の少なくとも一つは、条件付きまたは無条件ジャンプまたは分岐命令である。上述したような“スタックの一致（stack consistency）”を要求する結果として、各プログラムループ（program loop）によって、オペランドスタック内に格納されるオペランド数の正味の増加や減少が起こるようなことはなくなる。

【００５２】スタックスナップショット格納構造３４６は、スタックカウンタ３４２及び仮想スタック３４４の“スナップショット”を格納するのに用いられ、プログラムの様々な点に於ける仮想スタックステータスを効果的に比較することを可能としている。格納されるスタックスナップショットの各々は、例えば：　ＳＣ、ＤＴ１、ＤＴ２、ＤＴ３、．．．．、ＤＴｎのような形であり、ここでＳＣはスタックカウンタの値、ＤＴ１は仮想オペランドスタック内の最初のデータタイプ値、ＤＴ２は仮想オペランドスタック内の２番目のデータタイプ値であり、ＤＴｎまで同様である。ＤＴｎは仮想オペランドスタック内の、全アイテム中最後のアイテムに対するデータタイプ値である。

【００５３】スタックスナップショット格納構造３４６は、ディレクトリ部３４８とスナップショット部３５０の２つに分かれる。ディレクトリ部３４８はターゲット命令識別子（例えば、各ターゲット命令の絶対または相対アドレス）を格納するのに用いられ、スナップショット部３５０は、これらのターゲット命令識別子（target instruction identifier）に関連する仮想スタック３４４のスナップショットを格納するのに用いられる。“ターゲット”命令は、ジャンプまたは分岐命令の行き先となり得るバイトコード命令として定義される。例えば、条件付き分岐命令は、条件（満たされることも満たされないこともある）と、条件が成立したとき、処理が“ジャンプ”すべきプログラム内の場所（ターゲット）を示す分岐先支持部とを含む。条件付きジャンプ命令を評価する際、ベリファイヤ２４０は、スタックスナップショット格納構造３４６を用いて、ジャンプする直前の、ターゲット命令識別子をディレクトリ部３４８内に格納するとともに、仮想スタック３４４のステータスをスナップショット部３５０内に格納する。スタックスナップショット格納構造３４６の動作については、バイトコードベリファイヤプログラムの動作と共に、後により詳細に説明する。

【００５４】上述したように、バイトコードプログラム３４０は、データタイプに依存する複数の命令を含んでおり、それらの命令の各々は、本発明のベリファイヤ２４０によって評価される。バイトコードプログラム３５０は、スタック操作に関する命令３５２（スタックへの整数のプッシュ）及び３５４（スタックからの整数のポップ）、前方ジャンプ３５６とその関連するターゲット３６４、後方ジャンプ３６６とその関連するターゲット３６２、及びＤＯループ３５８とその終わり（end）３６０（ＤＯループのタイプによって、条件付きまたは無条件分岐命令であってもよい）を含んでいる。本発明の好適実施例のベリファイヤ２４０は、スタック操作及びデータタイプコンパチビリティをベリファイするだけなので、このバイトコードベリファイヤの動作は、これらの代表的な命令セットを用いることによって説明することができる。

【００５５】図６乃至図１２、及び付録２を参照して、バイトコードベリファイヤプログラム２４０の動作について以下に詳細に説明する。付録２は、ベリファイヤプログラムを疑似コードで表したものである。付録２の疑似コードは、本質的には、従来の汎用コンピュータ言語を用いたコンピュータ言語と同等である。この疑似コードは、本発明の説明のためにのみ作られたものであって、本分野の通常の知識を有するコンピュータプログラマであれば容易に理解されるように意図して書かれている。

【００５６】図６に示されているように、ダウンロードされたバイトコードプログラムは、処理のためバイトコードベリファイヤ２４０にロードされる（４００）。ベリファイヤ２４０は、オペランド及びローカル変数のデータタイプ情報を格納するため、メモリ内にアレイ状に位置を指定することによって、仮想スタック３４４及び仮想ローカル変数アレイ３４５を生成する（４０２）。同様に、ベリファイヤは、スナップショット情報を格納するため、メモリ内にアレイ上に位置を指定することによってスタックスナップショット格納構造を生成する（４０４）。最後に、ベリファイヤは、仮想スタックの

エントリ数を追跡するためのスタックカウンタ３４２として働くレジスタを指定する（４０６）。

【００５７】第１パスは、条件付き及び無条件ジャンプ及びループ命令に関連するターゲット情報を抽出するべく、バイトコードプログラム内に形成されたものである。第１パスに於いて、ベリファイヤ３００は全ての命令を順次処理していき（ステップ４０８、４１０、４１２）、命令が条件付きまたは無条件ジャンプ命令の場合（ステップ４１４）、かつそのジャンプに対するターゲット位置がまだディレクトリ３４８内に記憶されていない場合（ステップ４１８）、ターゲット位置を示す情報をスタックスナップショット格納構造３４６のディレクトリ部３４８に格納する（ステップ４１６）。例えば、ターゲット命令の絶対または相対アドレスを、ディレクトリ部３４８の次に使用可能なスロットに格納しても良い。他のタイプのバイトコード命令は、この第１パスでは無視される。

【００５８】プログラム内の全ての命令に対し処理が終了した後、ディレクトリ部３４８をソートして、リストアップされたターゲット位置をアドレス順に並び替えることが好ましい（４２０）。

【００５９】再度図５を参照されたい。図５では、説明のため、スタックスナップショット格納構造３４６は、既にベリフィケーションの第１パスがバイトコードプログラム３５０内に示されているバイトコード命令に基づいて終了したように、ディレクトリ部３４８内に情報が格納され、ロードされている。即ち、ディレクトリ部には、バイトコードプログラム内にある条件付き及び無条件ジャンプ命令の全てのターゲットのアドレスがロードされている。

【００６０】図７では、バイトコードプログラムによるオペランドスタック及びデータタイプの使用が適切かどうかをベリファイするため、バイトコードプログラムの第２のパスが開始される。バイトコードプログラムの最初の命令が選択されると（４３０）、ベリファイヤは、選択された命令に対するアドレスが、上述した第１パスに於いて既にスタックスナップショット格納構造３４６のディレクトリ部３４８内に格納されているかどうかチェックする（４３２）。

【００６１】選択された命令のアドレスがディレクトリ３４８内にある場合（即ち、選択された命令が条件付きまたは無条件ジャンプのターゲットであることを意味している）、ベリファイヤは関連するスタックスナップショットがスタックスナップショット格納構造３４６のスナップショット部３５０内に格納されているかどうかをチェックする（４３４）。スタックスナップショットが格納されていない場合（即ち、命令が後方ジャンプのターゲットであることを意味する）、スタックカウンタ及び仮想スタックの内容が、スタックスナップショット格納構造３４６内に格納される（４３６）。スナップショットは、処理中の命令を実行する直前の仮想スタックのステータスに関する情報を含み、それにはスタックにプッシュされている各データに対するデータタイプ値も含まれる。ベリファイヤは、その後ベリフィケーションプロセスを続け、後に述べるように、ステップ４５０からは個々の命令に対し解析を開始する。

【００６２】スタックスナップショットが、選択されている命令に対し格納されている場合（このターゲット命令に関連するジャンプ命令が既に処理されているということを意味する）、ベリファイヤは、スタックスナップショット格納構造３４６のスナップショット部３５０内に格納されている、この命令に対する仮想スタックスナップショット情報と、仮想スタックのそのときの状態とを比較する（４３８）。比較の結果、仮想スタックの状態とスナップショットとがマッチしないことが示された場合（スタックステータスミスマッチ）、スタックステータスミスマッチが発生したバイトコードプログラム内の場所を示すエラーメッセージまたは信号が生成される（４４０）。好適実施例では、仮想スタックとスナップショットが同数または同タイプのエントリを含んでいない場合にミスマッチが発生する。ベリファイヤは、その後、そのプログラムが“偽（False）”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（４４２）。プログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットすることにより、バイトコードインタープリタ２４２によるプログラムの実行が阻止される（図３）。

【００６３】選択されている命令に対し既に格納されているスタックスナップショットと、その時の仮想スタックの状態とがマッチする場合（４３８）、ベリファイヤはベリフィケーションプロセスを続け、後に述べるようにステップ４５０から個々の命令に対して分析を開始する。

【００６４】選択されている命令のアドレスがスタックスナップショット格納構造３４６のディレクトリ部３４８内に見つからない場合、またはスタックステータスミスマッチが検出されない場合、ベリファイヤは、その命令特有のスタック使用法及び機能に応じて、命令に対する一連のチェックのうち、いくつかを選択して実行する。

【００６５】図８を参照されたい。最初に行われるチェックは、オペランドスタックからデータをポップさせる命令に関する。選択されている命令によってデータがスタックからポップされる場合（４５０）、スタックカウンタが調べられ、命令のデータポップ要求を満足する十分なデータがスタック内にあるかどうかが判定される（４５２）。

【００６６】オペランドスタックがその命令に対し十分なデータを持っていない場合（４５２）、これはスタッ

クアンダーフローと呼ばれ、スタックアンダーフローが検出されたプログラム内の場所を示すエラー信号またはメッセージが生成される（４５４）。更にベリファイヤはプログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（４５６）。

【００６７】スタックアンダーフロー状態が検出されない場合、ベリファイヤは、予め仮想スタック内に格納されているデータタイプコード情報と、（もしあれば）選択されている命令のデータタイプ要求とを比較する。例えば、分析されるている命令の操作コード（opcode）がスタックからポップされた値の整数型の加算（integer add）を要求する場合、ベリファイヤは、ポップされている仮想スタックのアイテムのオペランド情報を比較して、それが適切なデータタイプであること、即ち整数であることを確認する。比較がマッチする結果となった場合、ベリファイヤは、ポップされているエントリに関連する情報を仮想スタックから削除し、仮想スタック３４４からポップされたエントリ数が反映されるようにスタックカウンタ３４２を更新する。

【００６８】仮想スタック３４４のポップされているエントリのオペランド情報と、選択されている命令のデータタイプ要求との間でミスマッチが検出された場合（４５８）は、ミスマッチがバイトコードプログラム内のどこで発生したかを示すメッセージが生成される（４６２）。更に、ベリファイヤは、プログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（４５６）。これで、ポップベリフィケーションプロセスが終了する。

【００６９】次に、図９を参照されたい。選択されている命令がデータをスタックにプッシュしようとする場合（４７０）、スタックカウンタが調べられ、その命令がプッシュしようとするデータを格納するための十分な余地がスタックにあるかどうかが判定される（４７２）。オペランドスタックに、その命令がプッシュしようとするデータを格納するための十分な余地がない場合（４７２）、これはスタックオーバフローと呼ばれ、スタックオーバフローが検出されたプログラム内の場所を示すエラー信号またはメッセージが生成される（４７４）。更に、ベリファイヤは、プログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（４７６）。

【００７０】スタックオーバフロー状態が検出されない場合、ベリファイヤは、選択されている命令によってスタックにプッシュされる各データに対し、（実際にプログラムが実行される際に）オペランドスタックにプッシュされるデータ（オペランド）のタイプを示すエントリを仮想スタックに加える（４７８）。この情報は、本発明の好適実施例のバイトコードプログラムで用いられている、データタイプが特定された操作コードから導かれる。また、ベリファイヤは、スタックカウンタ３４２を更新して、仮想スタックに追加された１または複数のエントリが反映されるようにする。これでスタックプッシュベリフィケーションプロセスが終了する。

【００７１】図１０を参照されたい。選択されている命令により、通常の逐次的ステップを飛び越すような、条件付きまたは無条件の前方ジャンプまたは分岐が発生する場合（４８０）、ベリファイヤは、まずそのジャンプ命令のターゲット位置に対するスナップショットがスタックスナップショット格納構造３４６内に格納されているかどうかをチェックする。スタックスナップショットが格納されていない場合、（そのジャンプに関連する仮想スタックの更新がなされた後の）仮想スタック状態が、スタックスナップショット格納構造３４６内の、そのターゲットプログラム位置に関連する位置に格納される（４８４）。このジャンプに関連するスタックポップ操作は、既に実行されているステップ４６０に於いて仮想スタック内に反映されていることに注意されたい（図８参照）。

【００７２】スタックスナップショットが格納されている場合（このターゲットに関連する他のエントリポイント（entry point）が既に処理されていることを示す）、ベリファイヤは、スタックスナップショット格納構造３４６のスナップショット部３５０内に格納されている仮想スタックスナップショット情報と、仮想スタックのその時の状態とを比較する。比較の結果、スナップショットと仮想スタックの状態とがマッチしていないことが示された場合（スタックステータスミスマッチ）、スタックステータスミスマッチが発生したバイトコードプログラム内の場所を示すエラーメッセージが生成される（４８８）。好適実施例では、スナップショットとその時の仮想スタックが、同数または同タイプのエントリを含んでいないときミスマッチが発生する。更に、その時の仮想スタック内の１以上のデータタイプ値が、スナップショット内の対応するデータタイプ値に一致しないときにもミスマッチが発生する。ベリファイヤは、その後プログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（４９０）。ステップ４８６でスタックステータスマッチングが検出された場合、ベリファイヤはステップ５００へと処理を継続する（図１１）。

【００７３】図１１を参照されたい。選択されている命令によってプログラムの後方に向かう条件付きまたは無条件ジャンプまたは分岐が発生する場合（ステップ５００）、ベリファイヤは、スタックスナップショット格納構造３４６のスナップショット部３５０内に格納されている、その後方ジャンプのターゲットに関連する仮想ス

タックスナップショット情報（ステップ４３６で格納した）と、仮想スタックのその時の状態とを比較する。比較の結果、スナップショットとその時のステータスがマッチしないことが示された場合（スタックステータスミスマッチ）、スタックステータスミスマッチが発生したバイトコードプログラム内の場所を示すエラーメッセージが生成される（５０４）。好適実施例では、ミスマッチは、その時の仮想スタックとスナップショットが同数または同タイプのエントリを含んでいないか、あるいは仮想スタック内のデータタイプエントリのどれかが、スナップショット内の対応するデータタイプエントリとマッチしない場合に発生する。ベリファイヤは、その後プログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（５０６）。

【００７４】スタックステータスがマッチしていることが検出されるか（ステップ５０２）、または命令が後方ジャンプでない場合（ステップ５００）、ベリファイヤはステップ５１０へと処理を継続する。

【００７５】選択されている命令が、ローカル変数からデータを読み出す場合（５１０）、ベリファイヤは対応する仮想ローカル変数内に予め格納されているデータタイプコード情報と、（もしあれば）選択されている命令のデータタイプ要求とを比較する。仮想ローカル変数に格納されているデータタイプ情報と選択されている命令のデータタイプ要求との間にミスマッチが検出された場合（５１２）、ミスマッチが発生したバイトコードプログラム内の場所を示すメッセージが生成される（５１４）。その後、ベリファイヤはプログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（５１６）。

【００７６】その時選択されている命令がローカル変数からデータを読み出さない場合（５１０）、またはステップ５１２に於けるデータタイプの比較がマッチする結果となった場合、ベリファイヤは、選択されている命令の処理を更にステップ５２０へと継続する。

【００７７】図１２を参照されたい。選択されている命令によってデータがローカル変数に格納される場合（５２０）、対応する仮想ローカル変数が調べられ、その仮想ローカル変数がデータタイプ値を格納しているかどうかが判定される（５２２）。その仮想ローカル変数がデータタイプ値を格納している場合（即ちそのデータは以前にそのローカル変数内に格納されたことを意味する）、ベリファイヤは、その仮想ローカル変数内にあるデータタイプ情報と、選択されているバイトコード命令に関するデータタイプとを比較する（５２４）。仮想ローカル変数内に格納されているデータタイプ情報と、選択されている命令によって要求されているデータタイプとの間にミスマッチが検出されると（５２４）、ミスマッチが発生したバイトコードプログラム内の場所を示すメッセージが生成される（５２６）。その後、ベリファイヤはプログラムが“偽”であることを示すようにベリフィケーションステータス２４５をセットし、ベリフィケーションプロセスをアボートする（５２８）。

【００７８】選択されている命令が、ローカル変数にデータを格納しない場合（５２０）、選択されている命令に対する処理は終了する。選択されている命令によってデータがローカル変数内に格納されるが、その仮想ローカル変数がデータタイプ値を格納していない場合（即ち、そのローカル変数にデータを格納するような命令がベリファイヤによってまだ処理されていないことを示す）、選択されているバイトコード命令に関連するデータタイプが、その仮想ローカル変数に格納される（ステップ５３０）。

【００７９】続いて、ベリファイヤは、その命令が処理中のバイトコードプログラム３４０の最後の命令であるかどうかをチェックする（５４０）。処理すべき命令がまだ残っている場合、ベリファイヤは次の命令をロードし（５４２）、ベリフィケーションプロセスをステップ４３２から繰り返す。それ以上処理する命令がない場合は、ベリファイヤはプログラムが“真”であることを示すようにベリフィケーションステータス２４５をセットし（５４４）、ベリフィケーションプロセスの終了を伝える。

【００８０】バイトコードインタープリタ

図１３のフローチャート及び付録３を参照しつつ、バイトコードインタープリタ２４２の動作について説明する。付録３は、バイトコードインタープリタを疑似コードで表したものである。

【００８１】指定されたバイトコードプログラムが与えられた場合、即ち、実行すべきプログラムとして選択された後（５６０）、バイトコードプログラムインタープリタ２４２は、指定されたバイトコードプログラムの保全性をベリファイするべく、バイトコードベリファイヤ２４０を呼び出す（５６２）。バイトコードベリファイヤについては既に上述した。

【００８２】ベリファイヤが“ベリフィケーション失敗（verification failure）”をリターンしてきた場合（５６４）、指定されたバイトコードプログラムのインタープリタによって実行されることなくアボートされる（５６６）。

【００８３】ベリファイヤ２４２が“ベリフィケーション成功”をリターンしてきた場合（５６４）、指定されたバイトコードプログラムはリソースユーティリティプログラムにリンクされ（５６８）、更に、そのプログラムによって参照される他のプログラム、関数、及びオブジェクトにもリンクされる。そのようなリンク過程は多くのプログラムインタープリタに於ける従来の前処理（pre-execution）過程と同様である。その後リンクさ

れたバイトコードプログラムはインタープリタによってインタープリットされ実行される（５７０）。本発明のバイトコードインタープリタは、プログラムの実行中、オペランドスタックオーバフロー及びアンダーフローのチェックをしない。また、オペランドスタック内に格納されているデータに対するデータタイプのチェックもしない。このような従来インタープリット過程で行われていたスタックオーバフロー／アンダーフロー、及びデータタイプのチェックは、本発明では省略することができる。これは、ベリファイヤによって、このようなエラーがプログラム実行中に発生しないことが既にベリファイされているためである。

【００８４】本発明のプログラムインタープリタは、何度も繰り返し実行される命令ループ（instruction loop）を有するバイトコードプログラムの実行に対し、特に効果的である。なぜなら、本発明では、そのような各命令ループ内の各バイトコードに対して、オペランドスタックチェック命令は一度だけしか行われないからである。対照的に、従来のインタープリタによるプログラムの実行に於いては、インタープリタは、オーバフロー（即ち、スタックが格納できるより多くのデータがスタックに加えられる）やアンダーフロー（即ち、スタックが空の時にスタックからデータをポップしようとする）が生じていないかどうか、オペランドスタックを連続的＊

＊にモニタする必要がある。そのようなスタックモニタリングは、通常、スタックのステータスを変化させる全ての命令（即ち、ほとんど全ての命令）に対して実行しなければならない。多くのプログラムで、インタープリタによって実行されるスタックモニタリング命令は、インタープリットされるコンピュータプログラムの実行時間の約８０％に達する。その結果、本発明のインタープリタは、しばしば、同じコンピュータ上で実行される従来のプログラムインタープリタの２倍から５倍の速さでプログラムを実行する。

【００８５】本発明の特定の実施例について行った上述の説明は、例示を目的としたものであり、本発明を網羅的に記載することを意図したものでも、本発明を開示した形態そのものに限定することを意図したものでもなく、上述したことから多くの変形変更が可能であることは明らかであろう。上記の実施例は、本発明の原理とその実際的な応用が最も良く説明されるように選択及び記述したものであり、それによって当業者が本発明を最も良く利用して、特定の使用に合うように様々な改良を加えた変形実施態様が可能なようにしたものである。本発明の範囲は、特許請求の範囲によって画定される。

【００８６】

【実施例】

表１

ＯＡＫ言語のバイトコード

命令の名前	簡単な説明
aaload	アレイからのオブジェクトリファレンスのロード
aastore	オブジェクトリファレンスのオブジェクトリファレンスアレイへの格納
aconst_null	ヌルオブジェクト（null object）のプッシュ
aload	ローカルオブジェクト変数のロード
areturn	関数からのオブジェクトリファレンスのリターン
arraylength	アレイ長の取得
astore	オブジェクトリファレンスのローカル変数への格納
astore_<n>	オブジェクトリファレンスのローカル変数への格納
athrow	例外の破棄
bipush	１バイト符号付整数のプッシュ
breakpoint	ブレイクポイントハンドラ（breakpoint handler）のコール
catchsetup	例外ハンドラ（exception handler）のセットアップ
catchteardown	例外ハンドラのリセット
checkcast	オブジェクトが所与のタイプであることの確認
df2	倍精度浮動小数点数の単精度浮動少数点数への変換
d2i	倍精度浮動小数点数の整数への変換
d2l	倍精度浮動小数点数の倍長整数への変換
dadd	倍精度浮動小数点数の加算
daload	倍精度浮動小数点数のアレイからのロード
dastore	倍精度浮動小数点数のアレイへの格納

27 28

dcmpg	２つの倍精度浮動小数点数の比較（比較できない場合１をリターン）
dcmpl	２つの倍精度浮動小数点数の比較（比較できない場合－１をリターン）
dconst_<d>	倍精度浮動小数点数のプッシュ
ddiv	倍精度浮動小数点数の除算
dload	倍精度浮動小数点数のローカル変数からのロード
dload_<n>	倍精度浮動小数点数のローカル変数からのロード
dmod	倍精度浮動小数点数のモジューロ関数の実行
dmul	倍精度浮動小数点数の乗算
dneg	倍精度浮動小数点数の無効化
dreturn	倍精度浮動小数点数の関数からのリターン
dstore	倍精度浮動小数点数のローカル変数への格納
dstore_<n>	倍精度浮動小数点数のローカル変数への格納
dsub	倍精度浮動小数点数の引き算
dup	一番上のスタックワードの複製
dup2	一番上の２つのスタックワードの複製
dup2_x1	一番上の２つのスタックワードを複製し２つ下げる
dup2_x2	一番上の２つのスタックワードを複製し３つ下げる
dup_x1	一番上のスタックワードを複製し２つ下げる
dup_x2	一番上のスタックワードを複製し３つ下げる
f2d	単精度浮動小数点数の倍精度浮動小数点数への変換
f2i	単精度浮動小数点数の整数への変換
f2l	単精度浮動小数点数の倍長整数への変換
fadd	単精度浮動小数点数の加算
faload	単精度浮動小数点数のアレイからのロード
fastore	単精度浮動小数点数のアレイへの格納
fempg	単精度浮動小数点数の比較（比較不能な場合１を返す）
fempl	単精度浮動小数点数の比較（比較不能な場合－１を返す）
fconst_<f>	単精度浮動小数点数のプッシュ
fdiv	単精度浮動小数点数の除算
fload	単精度浮動小数点数のローカル変数からのロード
fload_<n>	単精度浮動小数点数のローカル変数からのロード
fmod	単精度浮動小数点数のモジューロ関数の実行
fmul	単精度浮動小数点数の乗算
fneg	単精度浮動小数点数の無効化
freturn	単精度浮動小数点数の関数からのリターン
fstore	単精度浮動小数点数のローカル変数への格納
fstore_<n>	単精度浮動小数点数のローカル変数への格納
fsub	単精度浮動小数点数の引き算
getfield	オブジェクトからのフィールドのフェッチ
getstatic	クラスからのスタティックフィールドのセット
goto	無条件分岐
i2d	整数の倍精度浮動小数点数への変換
i2f	整数の単精度浮動小数点数への変換
i2l	整数の倍長整数への変換
iadd	整数の加算
iaload	整数のアレイからのロード

29 30

iand	２つの整数のブール代数ＡＮＤ
iastore	整数のアレイへの格納
iconst_<n>	整数のプッシュ
iconst_m1	整数定数から１を引いてプッシュ
idiv	整数の除算
if_acmpeq	オブジェクトが同じ場合分枝
if_acmpne	オブジェクトが同じでない場合分枝
if_icmpeq	整数が等しい場合分枝
if_icmpge	整数がある値以上の場合分枝
if_icmpgt	整数がある値より大きい場合分枝
if_icmple	整数がある値以下の場合分枝
if_icmplt	整数がある値未満の場合分枝
if_icmpne	整数がある値に等しくない場合分枝
ifeq	０に等しい場合分枝
ifge	０以上の場合分枝
ifgt	０より大きい場合分枝
ifle	０以下の場合分枝
iflt	０未満の場合分枝
ifne	０に等しくない場合分枝
iinc	定数分だけローカル変数を増加
iload	整数のローカル変数からのロード
iload_<n>	整数のローカル変数からのロード
imod	整数のモジューロ関数の実行
imul	整数の乗算
ineg	整数の無効化
instanceof	オブジェクトが所与のタイプであるかどうかの判定
int2byte	整数の符号付バイト（signed byte）への変換
int2char	整数のキャラクタ（char）への変換
invokeinterface	インタフェースメソッドの呼び出し
invokemethod	クラスメソッドの呼び出し
invokesuper	スーパークラスメソッドの呼び出し
ior	２つの整数のブール代数ＯＲ
ireturn	整数の関数からのリターン
ishl	整数の左シフト
ishr	整数の算術的右シフト（arithmetic shift right）
istore	整数のローカル変数vindexへの格納
istore_<n>	整数のローカル変数ｎへの格納
isub	整数の引き算
iushr	整数の論理右シフト
ixor	２つの整数のブール代数ＸＯＲ
jsr	サブルーチンへのジャンプ
l2d	倍長整数の倍精度浮動小数点数への変換
l2f	倍長整数の単精度浮動小数点数への変換
l2i	倍長整数の整数への変換
ladd	倍長整数の加算
laload	倍長整数のアレイからのロード
land	２つの倍長整数のブール代数ＡＮＤ
lastore	倍長整数のアレイへの格納
lcmp	倍長整数の比較
lconst_<l>	倍長整数定数のプッシュ

ldc1	定数プールからのアイテムのプッシュ
ldc2	定数プールからのアイテムのプッシュ
ldc2w	定数プールからのロングまたはダブルのプッシュ
ldiv	倍長整数の除算
lload	倍長整数のローカル変数からのロード
lload_<n>	倍長整数のローカル変数からのロード
lmod	倍長整数のモジューロ関数の実行
lmul	倍長整数の乗算
lneg	倍長整数の無効化
lookupswitch	キーマッチ（key match）によるジャンプテーブルへのアクセス及びジャンプ
lor	２つの倍長整数のブール代数ＯＲ
lreturn	倍長整数の関数からのリターン
lshl	倍長整数の左シフト
lshr	倍長整数の算術的右シフト
lstore	倍長整数のローカル変数への格納
lstore_<n>	倍長整数のローカル変数への格納
lsub	倍長整数の引き算
lushr	倍長整数の論理右シフト
lxor	倍長整数のブール代数ＸＯＲ
monitorenter	コードのモニタされている領域へのエンター
monitorexit	コードのモニタされている領域からの離脱
new	新たなオブジェクトの生成
newarray	新たなアレイの位置決め
newfromname	ネーム（name）からの新たなオブジェクトの生成
nop	何もしない
pop	１番上のスタックワードのポップ
pop2	１番上の２つのスタックワードのポップ
putfield	オブジェクト内のフィールドのセット
putstatic	クラス内のスタティックフィールドのセット
ret	サブルーチンからのリターン
return	プロシージャ（procedure）からのリターン（ボイド）
saload	符号付バイトのアレイからのロード
sastore	符号付バイトのアレイへの格納
siaload	符号なしショート（short）のアレイからのロード
siastore	符号なしショートのアレイへの格納
sipush	２バイト符号付整数のプッシュ
tableswitch	インデックスによるジャンプテーブルへのアクセス及びジャンプ
verifystack	スタックが空であることをベリファイする

【００８７】

付録１

クラスローダの疑似コード

見たいオブジェクト（“被参照オブジェクト”）のユーザによる選択（例えば、ある文献内のオブジェクトまたは他のオブジェクトへのハイパーリンクを選択することによって選択することが出来る）
被参照オブジェクトを格納しているサーバへの接続を開く
被参照オブジェクトに対するハンドル（データタイプを含む）の受け取り
データタイプがユーザシステムにとって既知であるかどうか（即ち、受け取ったデータタイプのオブジェクトに対応するビューワをユーザが持っているかどうか

```
) のチェック
データタイプが既知でない場合
  {
   同じサーバへの第２の接続を開く
   指定されたデータタイプに対応するビューワのリクエスト：
   成功の場合
     ／＊ハイブリッドシステムオプション：ノンバイトコードビューワ
     (non-bytecode viewer) に対するチェック＊／
     {
      受け取ったビューワがバイトコードプログラムでない場合
         {
          ビューワをアクセプトするか否かの決定
          ビューワがアクセプトされない場合
              受け取ったビューワの削除
          ビューワがアクセプトされる場合　／＊ノンバイトコードビューワ
          がアクセプトされる＊／
              FinishObjectDownloadへ行く
          }
     ／＊ベリフィケーション及びレジストレーションプロシージャ＊／
      受け取ったビューワがバイトコードプログラムの場合
         {
          受け取ったビューワに対するバイトコードベリファイヤの実行
          ベリフィケーション成功の場合
             {
              ／＊ビューワのレジストレーション＊／
              受け取ったビューワをベリファイ済みとしてマーキングする
              ビューワのローカルビューワライブラリへの格納
              既知データタイプリストに、そのデータタイプを加える
              }
          ベリフィケーションが不成功の場合
              受け取ったビューワの削除
          }
      }
   データタイプがなお認識されない場合
     {
      指定されたデータタイプに対応するビューワを他のサーバ中でサーチ
      成功の場合
         {上記と同じベリフィケーション及びレジストレーションプロシージ
          ャの実行}
      }
   データタイプがなお認識されない場合
     {
      被参照オブジェクトのダウンロードのアボート
      被参照オブジェクトに対応するビューワを見つけることができなかった
      ことをユーザに通知
      }
   }
FinishObjectDownload:　／＊ノンバイトコードビューワに対する分岐先＊／
被参照オブジェクトのダウンロードの完了
ダウンロードされたオブジェクトに組み込みバイトコードプログラムが含まれて
```

いる場合
{
組み込みプログラムに対するバイトコードベリファイヤの実行
ベリフィケーション成功の場合
組み込みプログラムをベリファイ済みとしてマーキングする
ベリフィケーション不成功の場合
{
受け取ったオブジェクトの削除
ダウンロードプロシージャのアボート
}
}
関連するデータタイプに対応するビューワを用いて被参照オブジェクトを見る

【００８８】

付録２

ＯＡＫバイトコードベリファイヤの疑似コード

```
ベリファイするバイトコードプログラムの受け取り
スタックステータス情報を格納するための仮想オベランドスタックデータ構造の
生成及びローカル変数データタイプ情報を格納するための仮想ローカル変数アレ
イの生成
仮想スタックスナップショットを格納するためのデータ構造の生成
バイトコードプログラムを通る第１パス：
    条件付き及び無条件ジャンプまたは分岐のターゲットとなる（即ち、２以上
    の先行命令からエンターされ得る）命令を全て見つける
    仮想スタックスナップショットデータ構造にそれらのターゲット命令のリス
    トを格納する
バイトコードプログラムを通る第２パス：
    VerificationSuccessを“真”にセットする
    最後のバイトコード命令を処理するまで繰り返す（ＤＯループの開始）：
      {
       次のバイトコード命令を（プログラム内の順番通りに）選択する
       命令がターゲット命令のリスト内にある場合
          {
           この命令に対する仮想スタックのスナップショットが既に存在する
           場合
              {
               仮想スタックの現在のステータスを格納されているスナップシ
               ョットと比較する
               スナップショットが現在の仮想スタックステータスとマッチし
               ない場合
                  {
                   スタックミスマッチが発生したプログラム内の場所を示す
                   メッセージをプリントする
                   ベリフィケーションをアボートする
                   VerificationSuccessを“偽”にセットする
                   リターン
                   }
              }
           この命令に対する仮想スタックのスナップショットが存在しない場
           合
               現在の仮想スタックステータスのスナップショットを格納する
```

```
    }
  命令のタイプによる場合分け：
    {
     命令によってデータがオペランドスタックからポップされる場合
       {
        スタックアンダーフローのチェック
        スタックアンダーフローの場合
          {
           アンダーフローが発生したプログラム内の場所を示すメッ
           セージをプリントする
           ベリフィケーションをアボートする
           リターン
           }
        スタックからポップされる各オペランドのデータタイプをバイ
        トコード命令によって要求されているデータタイプと比較する
        タイプミスマッチの場合
          {
           データタイプミスマッチが発生したプログラム内の場所を
           示すメッセージをプリントする
           VerificationSuccessを“偽”にセットする
           }
        ポップされるオペランドに対する情報を仮想スタックから削除
        する
        スタックカウンタを更新する
        }
     命令によってデータがオペランドスタックにプッシュされる場合
       {
        スタックオーバフローのチェック
        スタックオーバフローの場合
          {
           オーバフローが発生したプログラム内の場所を示すメッセ
           ージをプリントする
           ベリフィケーションをアボートする
           VerificationSuccessを“偽”にセットする
           リターン
           }
        オペランドスタックにプッシュされるデータのデータタイプを
        示す情報を仮想スタックに加える
        スタックカウンタを更新する
        }
     命令が前方ジャンプまたは分岐命令の場合
       {
        そのターゲット命令に対する仮想スタックのスナップショット
        が既に存在する場合
          {
           仮想スタックの現在のステータスを格納されているスナッ
           プショットと比較する
           スナップショットが現在の仮想スタックステータスとマッ
           チしない場合
             {
```

スタックミスマッチが発生したプログラム内の場所を
示すメッセージをプリントする
ベリフィケーションをアボートする
VerificationSuccessを“偽”にセットする
リターン
}
}
そのターゲット命令に対する仮想スタックのスナップショット
が存在しない場合
現在の仮想スタックステータスのスナップショットをその
ターゲット命令に対するスナップショットとして格納する
}

命令がループの終わりの後方ジャンプ命令或いは他の後方ジャンプ
または分岐命令の場合
{
現在のスタックステータスをターゲット命令に対して格納され
ているスナップショットと比較する
現在の仮想スタックステータスが格納されているスナップショ
ットとマッチしない場合
{
スタックミスマッチが発生したプログラム内の場所を示す
メッセージをプリントする
ベリフィケーションをアボートする
VerificationSuccessを“偽”にセットする
リターン
}
}
命令によってローカル変数からデータが読み出される場合
{
ローカル変数から読み出される各データタイプをバイトコード
命令によって必要とされるデータタイプと比較する
タイプミスマッチの場合
{
データタイプミスマッチが発生したプログラム内の場所を
示すメッセージをプリントする
VerificationSuccessを“偽”にセットする
}
}
命令によってローカル変数内にデータが格納される場合
{
対応する仮想ローカル変数に既にデータタイプ値が格納されて
いる場合
{
仮想ローカル変数に格納されているデータタイプ値を対応
するローカル変数に格納されるデータのデータタイプ（現
在のバイトコード命令によって扱われるデータタイプによ
って決定される）と比較する
タイプミスマッチの場合
{

41　　42

データタイプミスマッチが発生したプログラム内の場所を示すメッセージをプリントする
VerificationSuccessを"偽"にセットする
}
}
対応する仮想ローカル変数にデータタイプ値が格納されていない場合
対応するローカル変数に格納されるデータのデータタイプを示す情報を仮想ローカル変数に加える
}
} ／＊場合分けの終わり＊／
} ／＊ＤＯループの終了＊／
リターン（ベリフィケーション成功）

【００８９】

付録３

バイトコードインタープリタの疑似コード

実行すべき、指定されたバイトコードプログラムの受け取り
指定されたバイトコードプログラムをベリファイするためのバイトコードベリファイヤのコール
ベリフィケーション成功の場合
{
指定されたバイトコードプログラムをリソースのユーティリティプログラムにリンクする
指定されたバイトコードプログラムの命令を、オペランドスタックオーバフロー及びアンダーフローに対するチェックを行うことなく、またオペランドスタックに格納されているオペランドに対するデータタイプのチェックを行うことなくインタープリットし、実行する
}

【図面の簡単な説明】

【図１】図１は、ネットワークを介して接続された２つのコンピュータを示している。

【図２】図２は、ネットワークを介して接続された２つのコンピュータを示しており、それらのコンピュータのうち少なくとも一つは、様々な実行可能形式のソースプログラムの複数のコピーを格納するための補助記憶装置（ディスク記憶装置）を含んでいる。

【図３】図３は、ネットワークを介して接続された２つのコンピュータを示しており、それらのコンピュータのうち少なくとも一つは、本発明によるバイトコードプログラムベリファイヤとクラスローダを含んでいる。

【図４】図４は、本発明の好適実施例に従って、リモートサーバ内にあるビューワとバイトコードプログラムにアクセスするためのロードプロセスのフローチャートである。

【図５】図５は、本発明によるバイトコードプログラムのベリフィケーションに於いてバイトコードベリファイヤによって維持されるデータ構造を示している。

【図６】図６は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図７】図７は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図８】図８は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図９】図９は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図１０】図１０は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図１１】図１１は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図１２】図１２は、本発明の好適実施例に於けるバイトコードプログラムベリフィケーションプロセスのフローチャートの一部である。

【図１３】図１３は、本発明の好適実施例に於けるバイトコードプログラムインタープリットプロセスのフローチャートである。

【符号の説明】

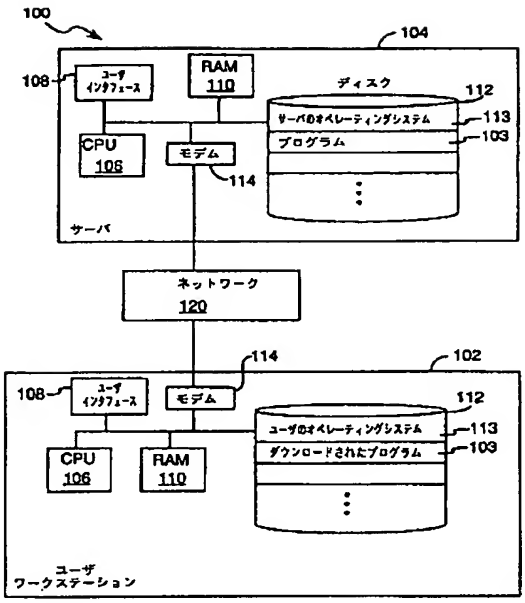
43

１００　従来の分散型コンピュータシステム
１０２　第１コンピュータ
１０３　コンピュータプログラム
１０４　第２コンピュータ
１０６　ＣＰＵ
１０８　ユーザインタフェース
１１０　主メモリ（ＲＡＭ）
１１２　補助記憶装置
１１３　オペレーティングシステム
１１４　コミニュケーションインタフェース（モデム）
１２０　コンピュータネットワーク
２００　従来の分散型コンピュータシステム
２０２　第１コンピュータノード（ユーザワークステーションノード）
２０４　第２コンピュータノード（サーバノード）
２０６　ＣＰＵ
２０８　ユーザインタフェース
２１０　主メモリ（ＲＡＭ）
２１２　補助記憶装置（ディスク記憶装置）
２１４　コミュニケーションインタフェース（モデム）
２１６　コンピュータコミュニケーションネットワーク
２１８　ＣＰＵ
２２０　ユーザインタフェース
２２２　主メモリ（ＲＡＭ）
２２４　補助記憶装置（ディスク記憶装置）
２２６　コミュニケーションインタフェース（モデム）
２２８　ディスクディレクトリ
２３０　オブジェクト
２３２　ビューワライブラリ
２３４　プログラム
２３６　オペレーティングシステム
２３７　オペレーティングシステム
２３８　プログラム
２３９　プログラム
２４０　バイトコードプログラムベリファイヤ
２４１　ビューワ
２４２　バイトコードプログラムインタープリタ
２４３　ビューワ
２４４　バイトコードプログラムコンパイラ
２４５　ベリフィケーションステータス
２５０　本発明による分散型コンピュータシステム
２５２　第１コンピュータノード（ユーザワークステーションノード）
２５４　第２コンピュータノード（サーバノード）

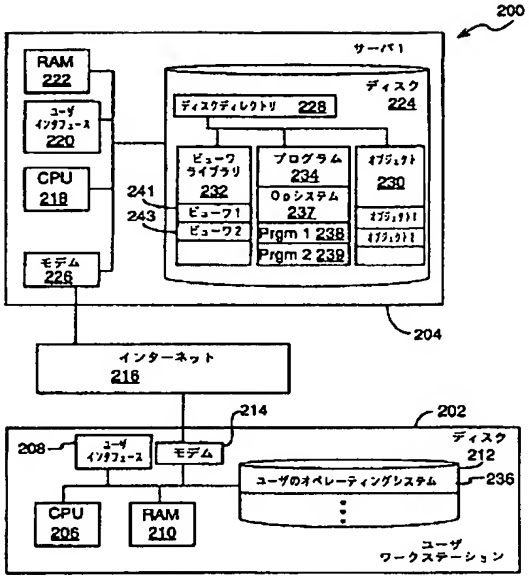
44

２５５　オペレーティングシステム
２５６　オペレーティングシステム
２５７　ＣＰＵ
２５８　ユーザインタフェース
２６０　主メモリ（ＲＡＭ）
２６２　補助記憶装置（ディスク記憶装置）
２６４　コミュニケーションインタフェース（モデム）
２６６　コンピュータコミュニケーションネットワーク（インターネット）
２６７　バイトコードプログラム
２６８　ＣＰＵ
２７０　ユーザインタフェース
２７２　主メモリ（ＲＡＭ）
２７４　補助記憶装置（ディスク記憶装置）
２７６　コミュニケーションインタフェース（モデム）
２８０　ディスクディレクトリ
２８２　オブジェクト
２８３　オブジェクト
２８４　ビューワライブラリ
２８６　プログラム
２９６　クラスローダプログラム
２９８　ビューワライブラリ
３０２　ベリフィケーションステータス
３２４　第２サーバ
３２５　補助記憶装置（ディスク記憶装置）
３２７　認識されているサーバのリスト
３２６　ビューワライブラリ
３４０　バイトコードプログラム
３４２　スタックカウンタ
３４４　仮想スタック
３４５　仮想ローカル変数アレイ
３４６　スタックスナップショット格納構造
３４８　ディレクトリ部
３５０　スナップショット部
３５２　バイトコード命令
３５４　バイトコード命令
３５６　バイトコード命令
３５８　バイトコード命令
３６０　バイトコード命令
３６２　バイトコード命令
３６４　バイトコード命令
３６６　バイトコード命令
３６８　バイトコード命令

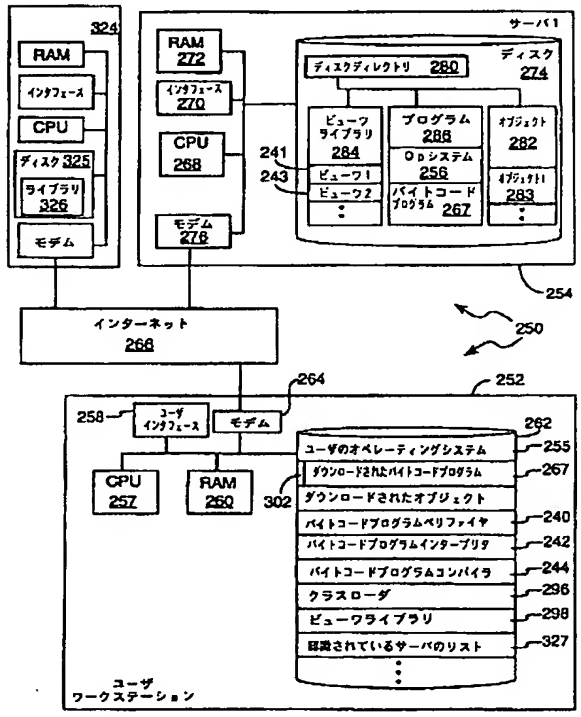
【図１】


100
104
108
RAM
110
ディスク
112
CPU
106
モデム
114
サーバ
113
103
ネットワーク
120
102
ユーザ
ワークステーション

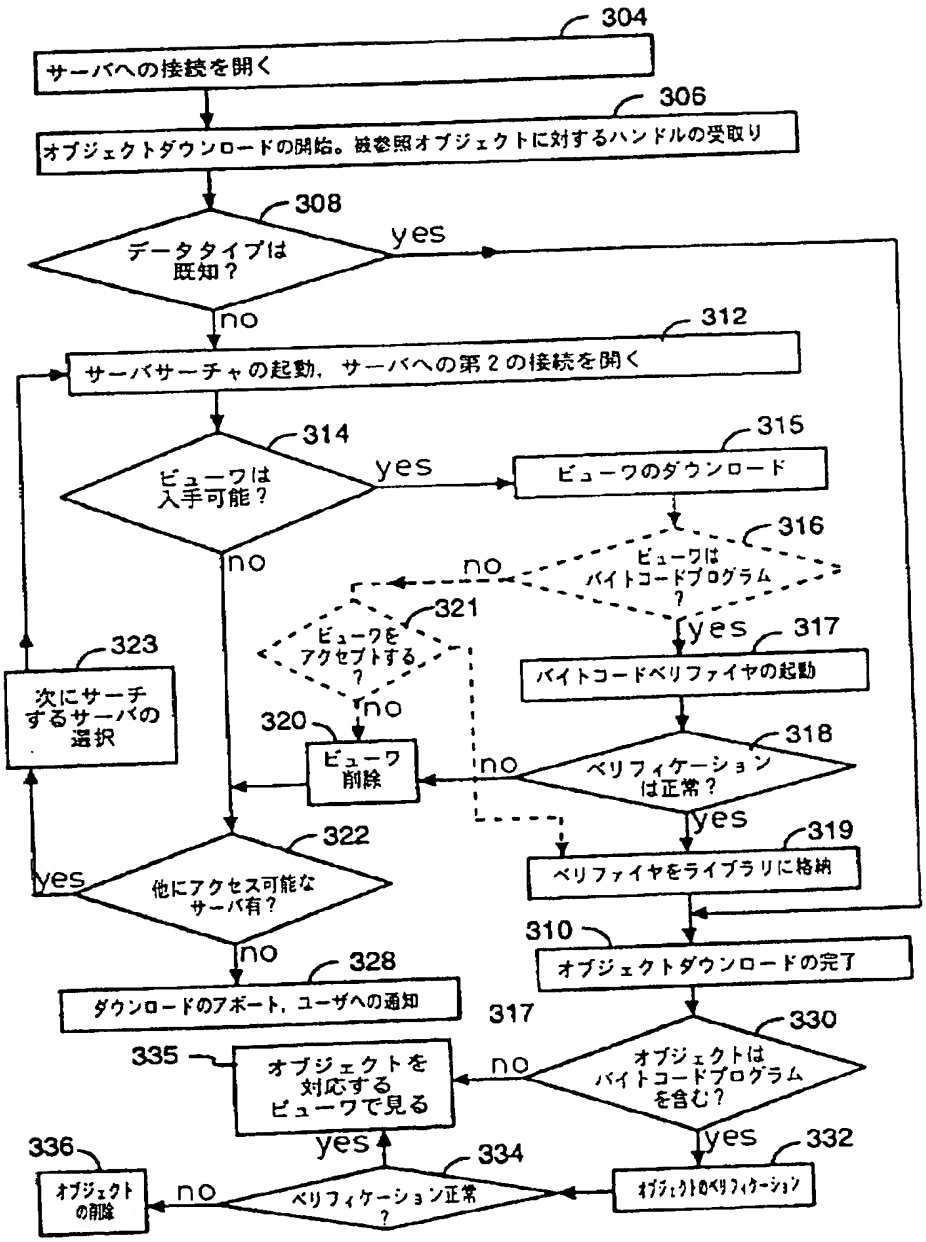
【図２】


200
サーバ1
RAM
222
ディスク
224
CPU
218
モデム
226
241
243
ビューワ1
ビューワ2
Prgm 1 238
Prgm 2 239
204
インターネット
216
214
202
208
CPU
206
RAM
210
ディスク
212
236
ユーザ
ワークステーション

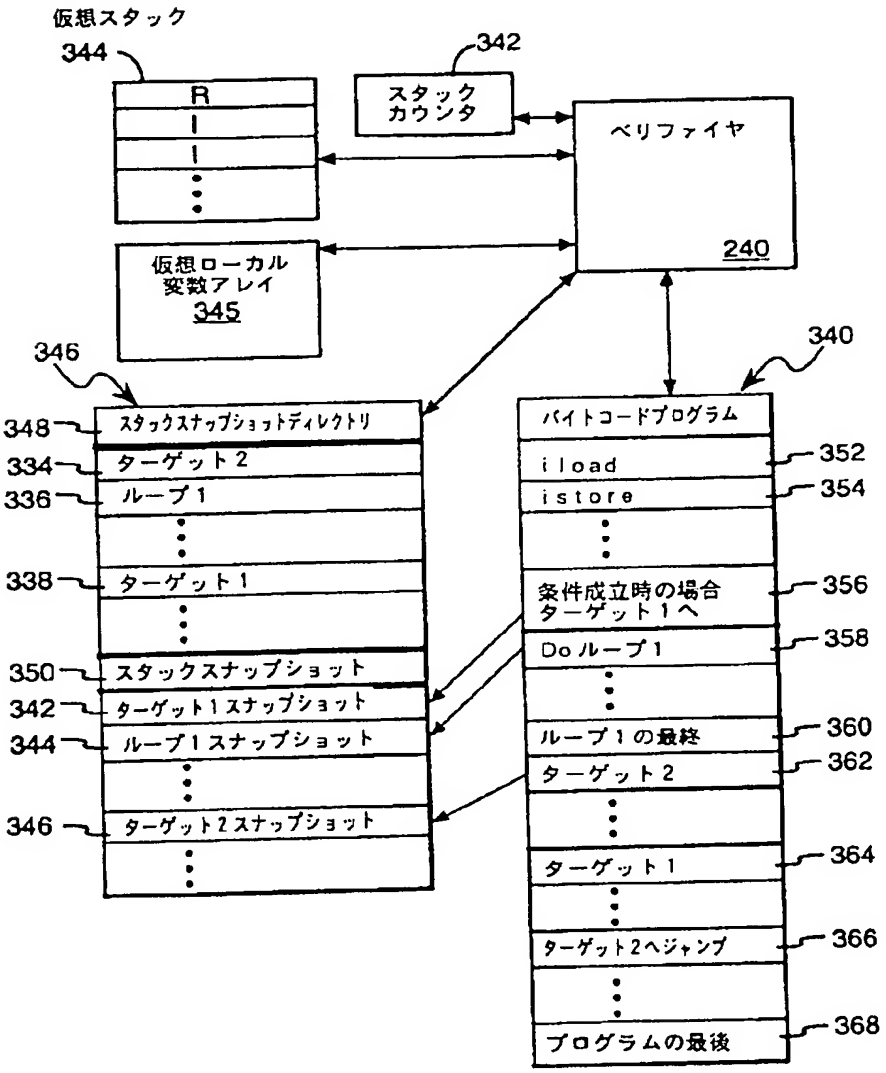
【図３】


324
RAM
CPU
ディスク 325
ライブラリ
326
モデム
サーバ1
RAM
272
CPU
268
モデム
278
ディスク
274
241
243
ビューワ1
ビューワ2
プログラム
286
Opシステム
256
バイトコード
プログラム 267
オブジェクト
282
オブジェクト1
283
254
250
インターネット
266
264
252
258
CPU
257
RAM
260
302
262
ユーザのオペレーティングシステム
255
ダウンロードされたバイトコードプログラム
267
ダウンロードされたオブジェクト
バイトコードプログラムベリファイヤ
240
バイトコードプログラムインタープリタ
242
バイトコードプログラムコンパイラ
244
クラスローダ
296
ビューワライブラリ
298
327
ユーザ
ワークステーション

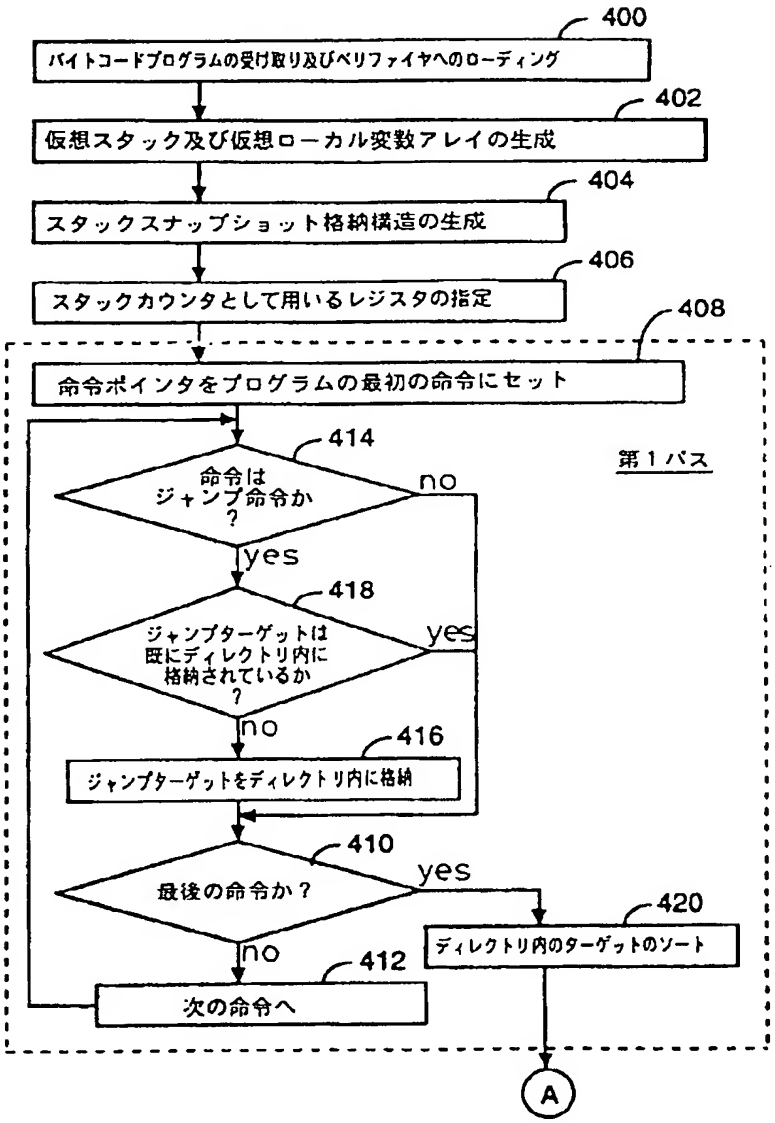
【図４】


304
サーバへの接続を開く
306
オブジェクトダウンロードの開始。被参照オブジェクトに対するハンドルの受取り
308
データタイプは既知？
yes
no
312
サーバサーチャの起動，サーバへの第２の接続を開く
314
ビューワは入手可能？
yes
315
ビューワのダウンロード
316
ビューワはバイトコードプログラム？
no
yes
321
ビューワをアクセプトする？
no
317
バイトコードベリファイヤの起動
323
次にサーチするサーバの選択
320
ビューワ削除
318
ベリフィケーションは正常？
no
yes
322
他にアクセス可能なサーバ有？
yes
no
319
ベリファイヤをライブラリに格納
310
オブジェクトダウンロードの完了
328
ダウンロードのアボート，ユーザへの通知
317
330
オブジェクトはバイトコードプログラムを含む？
335
オブジェクトを対応するビューワで見る
no
yes
yes
332
336
オブジェクトの削除
no
334
ベリフィケーション正常？

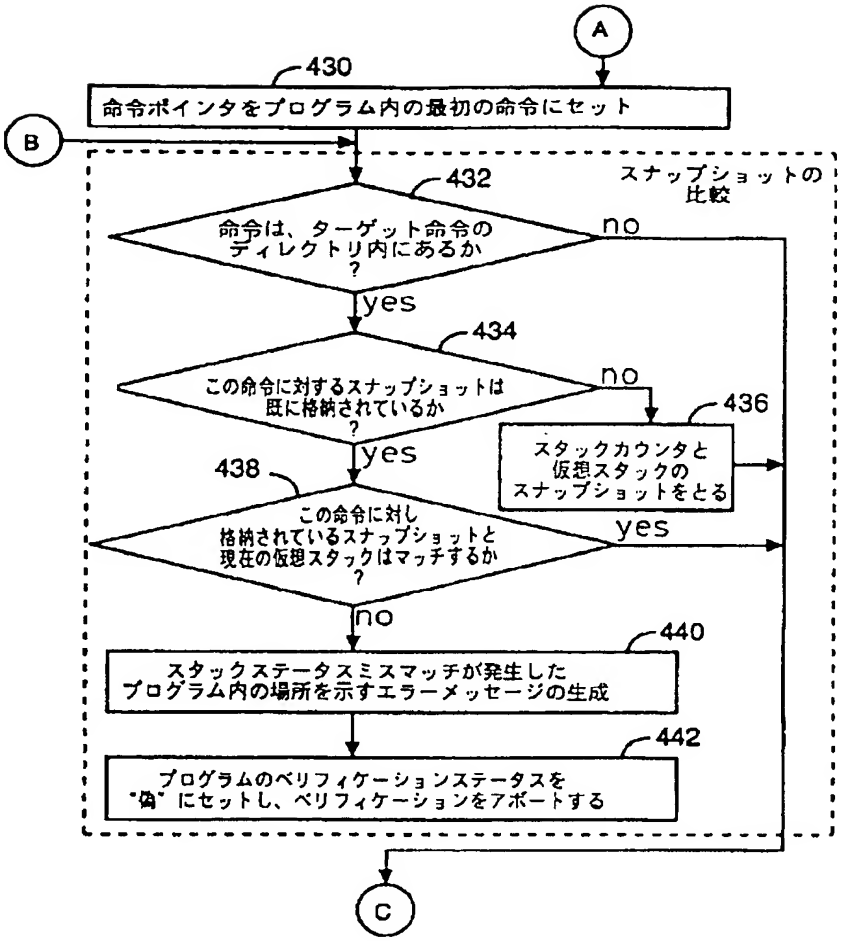
【図５】


仮想スタック
344
R
342
スタック
カウンタ
ベリファイヤ
240
仮想ローカル
変数アレイ
345
346
340
348
スタックスナップショットディレクトリ
334
ターゲット２
336
ループ１
338
ターゲット１
350
スタックスナップショット
342
ターゲット１スナップショット
344
ループ１スナップショット
346
ターゲット２スナップショット
バイトコードプログラム
352
iload
354
istore
356
条件成立時の場合
ターゲット１へ
358
Do ループ１
360
ループ１の最終
362
ターゲット２
364
ターゲット１
366
ターゲット２へジャンプ
368
プログラムの最後

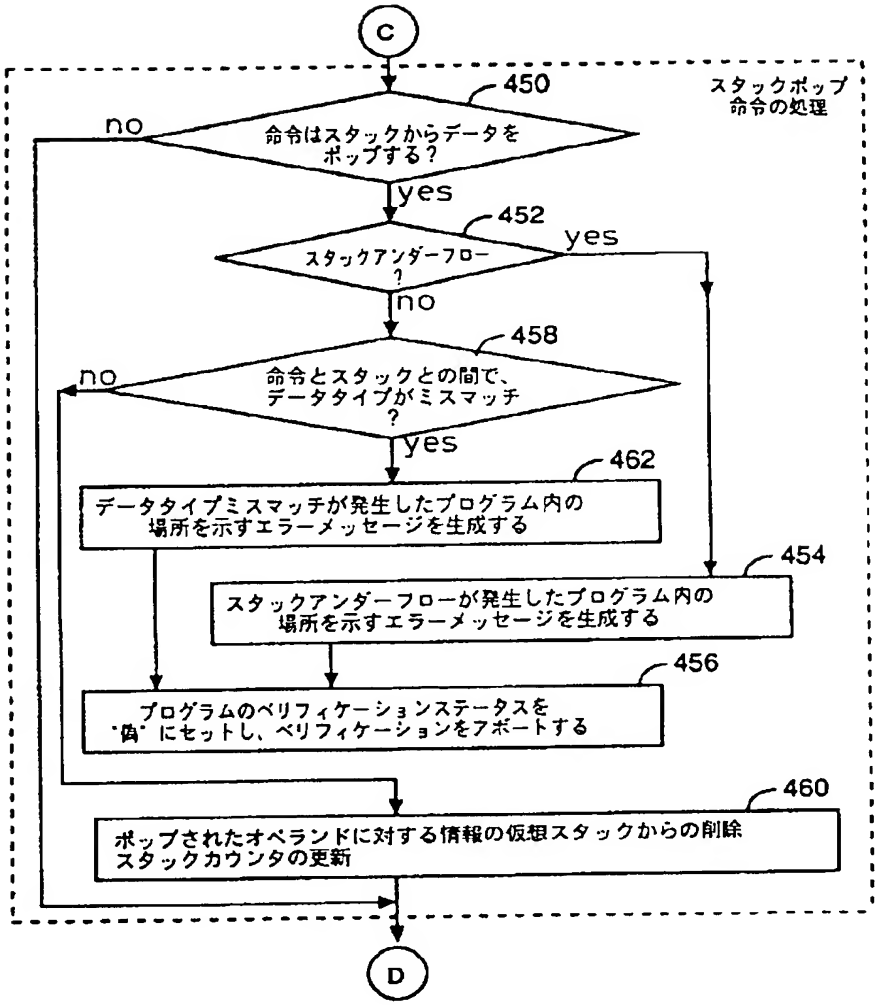
【図６】


400
バイトコードプログラムの受け取り及びベリファイヤへのローディング
402
仮想スタック及び仮想ローカル変数アレイの生成
404
スタックスナップショット格納構造の生成
406
スタックカウンタとして用いるレジスタの指定
408
命令ポインタをプログラムの最初の命令にセット
第１パス
414
命令は
ジャンプ命令か
？
no
yes
418
ジャンプターゲットは
既にディレクトリ内に
格納されているか
？
yes
no
416
ジャンプターゲットをディレクトリ内に格納
410
最後の命令か？
yes
420
ディレクトリ内のターゲットのソート
no
412
次の命令へ
A

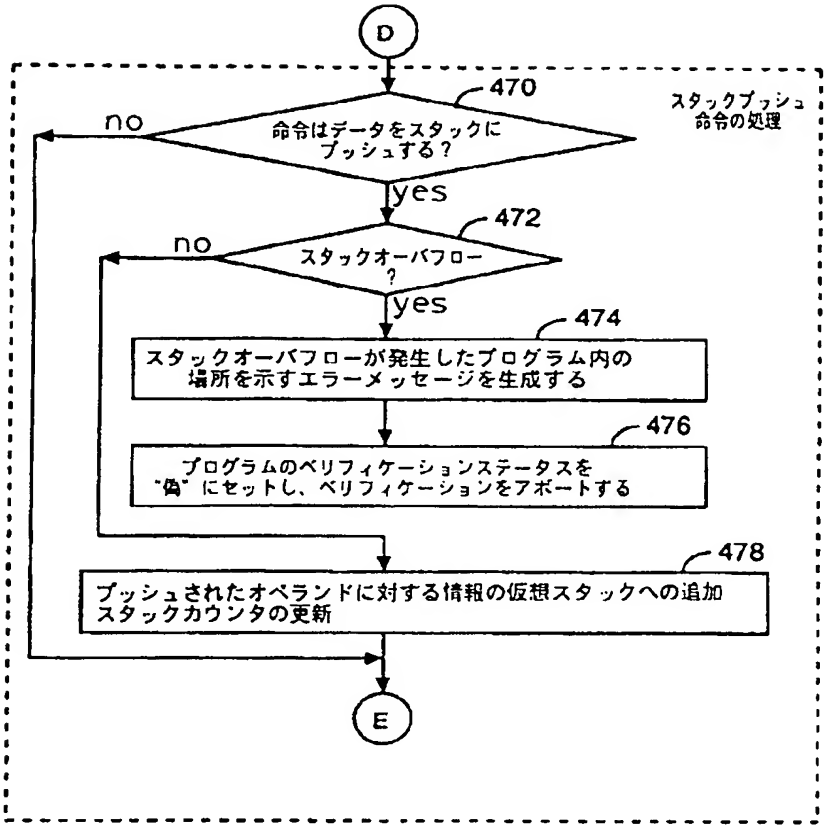
【図７】


A
430
命令ポインタをプログラム内の最初の命令にセット
B
スナップショットの比較
432
命令は、ターゲット命令のディレクトリ内にあるか？
no
yes
434
この命令に対するスナップショットは既に格納されているか？
no
436
スタックカウンタと仮想スタックのスナップショットをとる
yes
438
この命令に対し格納されているスナップショットと現在の仮想スタックはマッチするか？
yes
no
440
スタックステータスミスマッチが発生したプログラム内の場所を示すエラーメッセージの生成
442
プログラムのベリフィケーションステータスを"偽"にセットし、ベリフィケーションをアボートする
C

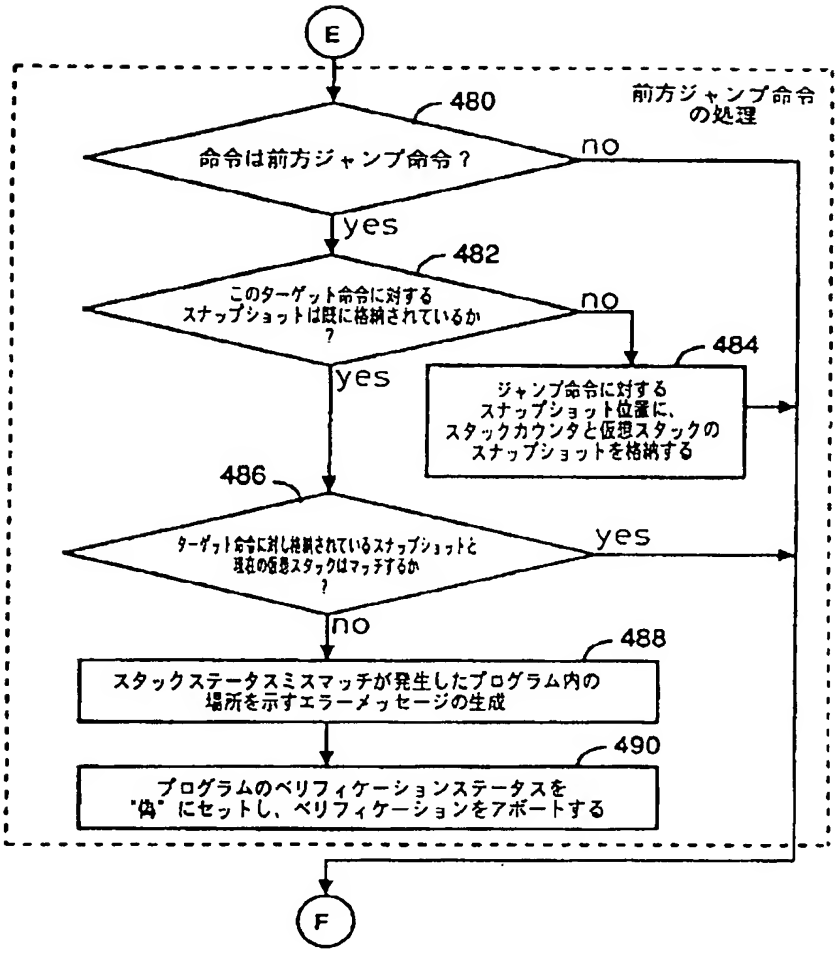
【図８】


C
スタックポップ
命令の処理
450
命令はスタックからデータを
ポップする？
no
yes
452
スタックアンダーフロー
？
yes
no
458
命令とスタックとの間で、
データタイプがミスマッチ
？
no
yes
462
データタイプミスマッチが発生したプログラム内の
場所を示すエラーメッセージを生成する
454
スタックアンダーフローが発生したプログラム内の
場所を示すエラーメッセージを生成する
456
プログラムのベリフィケーションステータスを
"偽" にセットし、ベリフィケーションをアボートする
460
ポップされたオペランドに対する情報の仮想スタックからの削除
スタックカウンタの更新
D

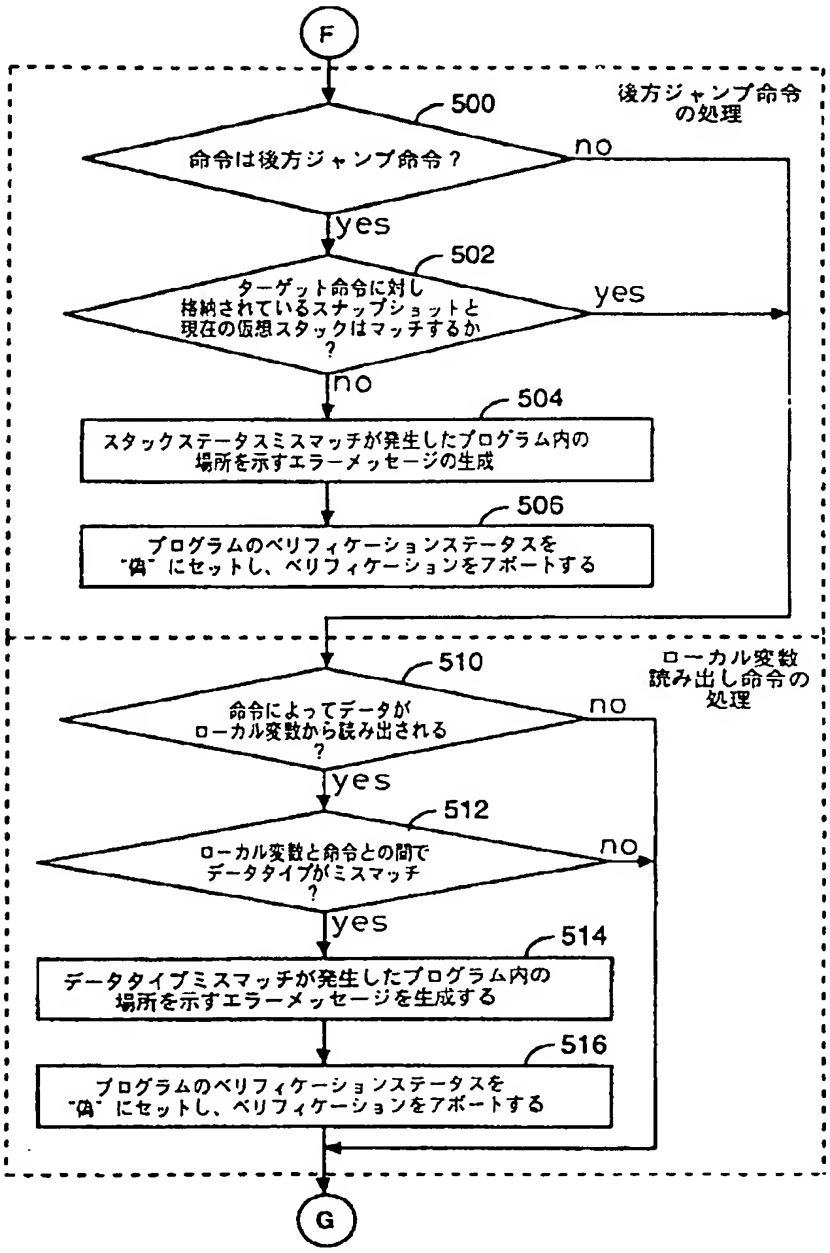
【図９】


D
470
スタックプッシュ
命令の処理
no
命令はデータをスタックに
プッシュする？
yes
472
no
スタックオーバフロー
？
yes
474
スタックオーバフローが発生したプログラム内の
場所を示すエラーメッセージを生成する
476
プログラムのベリフィケーションステータスを
"偽" にセットし、ベリフィケーションをアボートする
478
プッシュされたオペランドに対する情報の仮想スタックへの追加
スタックカウンタの更新
E

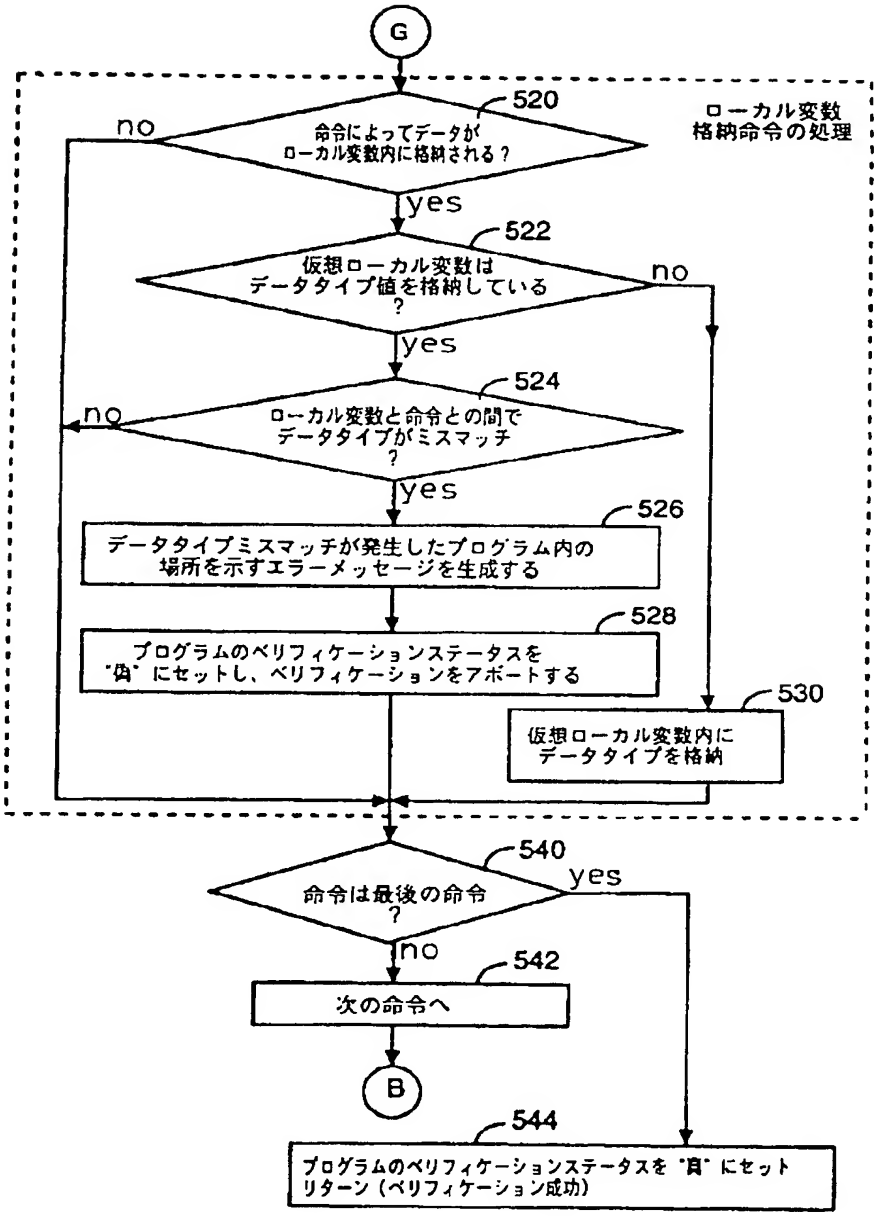
【図１０】


E
前方ジャンプ命令の処理
480
命令は前方ジャンプ命令？
no
yes
482
このターゲット命令に対するスナップショットは既に格納されているか？
no
484
ジャンプ命令に対するスナップショット位置に、スタックカウンタと仮想スタックのスナップショットを格納する
yes
486
ターゲット命令に対し格納されているスナップショットと現在の仮想スタックはマッチするか？
yes
no
488
スタックステータスミスマッチが発生したプログラム内の場所を示すエラーメッセージの生成
490
プログラムのベリフィケーションステータスを"偽"にセットし、ベリフィケーションをアボートする
F

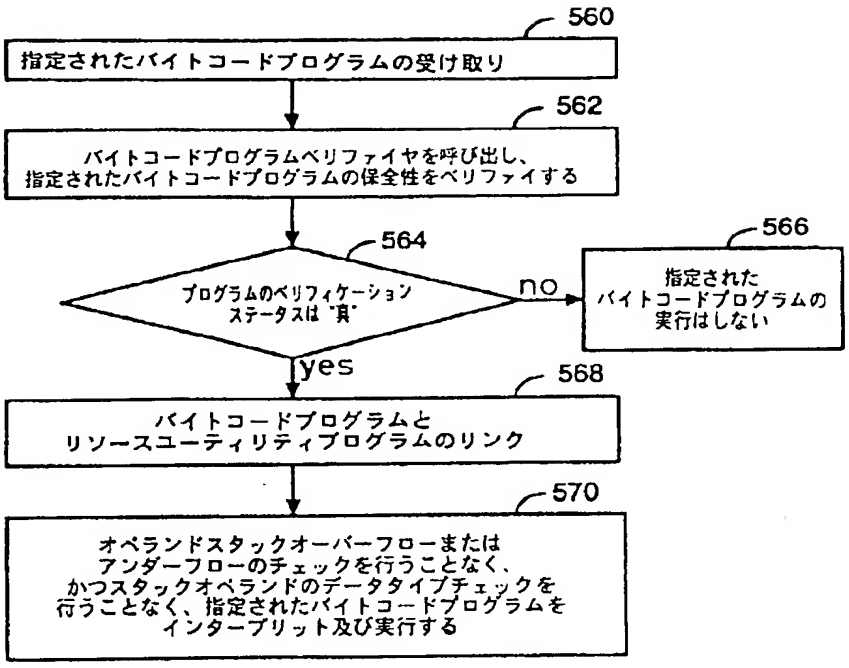
【図１１】


F
後方ジャンプ命令の処理
500
命令は後方ジャンプ命令？
no
yes
502
ターゲット命令に対し格納されているスナップショットと現在の仮想スタックはマッチするか？
yes
no
504
スタックステータスミスマッチが発生したプログラム内の場所を示すエラーメッセージの生成
506
プログラムのベリフィケーションステータスを"偽"にセットし、ベリフィケーションをアボートする
ローカル変数読み出し命令の処理
510
命令によってデータがローカル変数から読み出される？
no
yes
512
ローカル変数と命令との間でデータタイプがミスマッチ？
no
yes
514
データタイプミスマッチが発生したプログラム内の場所を示すエラーメッセージを生成する
516
プログラムのベリフィケーションステータスを"偽"にセットし、ベリフィケーションをアボートする
G

【図１２】


G
520
命令によってデータが
ローカル変数内に格納される？
no
yes
ローカル変数
格納命令の処理
522
仮想ローカル変数は
データタイプ値を格納している
？
no
yes
524
ローカル変数と命令との間で
データタイプがミスマッチ
？
no
yes
526
データタイプミスマッチが発生したプログラム内の
場所を示すエラーメッセージを生成する
528
プログラムのベリフィケーションステータスを
"偽" にセットし、ベリフィケーションをアボートする
530
仮想ローカル変数内に
データタイプを格納
540
命令は最後の命令
？
yes
no
542
次の命令へ
B
544
プログラムのベリフィケーションステータスを "真" にセット
リターン（ベリフィケーション成功）

【図１３】


560
指定されたバイトコードプログラムの受け取り
562
バイトコードプログラムベリファイヤを呼び出し、
指定されたバイトコードプログラムの保全性をベリファイする
564
プログラムのベリフィケーション
ステータスは "真"
no
566
指定された
バイトコードプログラムの
実行はしない
yes
568
バイトコードプログラムと
リソースユーティリティプログラムのリンク
570
オペランドスタックオーバーフローまたは
アンダーフローのチェックを行うことなく、
かつスタックオペランドのデータタイプチェックを
行うことなく、指定されたバイトコードプログラムを
インタープリット及び実行する

【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１５年４月１１日（２００３．４．１１）

【公開番号】特開平８－２６３４４７
【公開日】平成８年１０月１１日（１９９６．１０．１１）
【年通号数】公開特許公報８－２６３５
【出願番号】特願平７－３４９１６４
【国際特許分類第７版】
G06F 15/16 370
9/445
13/00 351
【ＦＩ】
G06F 15/16 370 N
13/00 351 H
9/06 420 J

【手続補正書】
【提出日】平成１４年１２月１９日（２００２．１２．１９）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】発明の名称
【補正方法】変更
【補正内容】
【発明の名称】　分散型コンピュータシステムの動作方法及びコンピュータシステムに関連するデバイス及びコンピュータシステムプロダクト
【手続補正２】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正内容】
【特許請求の範囲】
【請求項１】　複数の個別のコンピュータを有する分散型コンピュータシステムの動作方法であって、
（ａ）第１コンピュータに於いて、ビューワプログラムの第１ライブラリを格納する過程であって、各ビューワプログラムを以ってそのユーザが対応するデータタイプのオブジェクトを見ることを可能にする該過程と、
（ｂ）前記第２コンピュータに於いて、オブジェクト及びビューワプログラムの第２ライブラリを格納する過程であって、ビューワプログラムの前記第１及び第２ライブラリに於ける各ビューワプログラムを以って前記第２コンピュータのユーザが対応するデータタイプのオブジェクトを見ることを可能にする該過程と、
（ｃ）前記第２コンピュータに於いて、前記第２コンピュータの前記ユーザがオブジェクトを選択することを可能にし、前記選択されたオブジェクトの対応するデータタイプを判定する過程と、
（ｄ）前記第２コンピュータに於いて、前記第２コンピュータに格納された前記ビューワプログラムの中に、前記選択されたオブジェクトのデータタイプに対応するビューワプログラムが含まれているかどうか判定する過程と、
（ｅ）前記過程（ｄ）での判定結果が否定の場合、前記選択されたオブジェクトのデータタイプに対応するビューワプログラムが前記第１コンピュータに格納されているかどうか判定する過程と、
（ｆ）前記過程（ｅ）での判定結果が肯定の場合、前記選択されたオブジェクトのデータタイプに対応する前記ビューワプログラムのコピーを前記第２コンピュータにロードする過程とを含むことを特徴とする方法。
【請求項２】　前記過程（ｅ）での判定結果が否定の場合、前記第２データタイプに対応するビューワプログラムが所定の他のコンピュータのセットの何れかの中に格納されているかどうか判定し、前記判定結果が肯定の場合、前記過程（ｆ）を実行する過程を更に含むことを特徴とする請求項１に記載の方法。
【請求項３】　複数の個別のコンピュータを有する分散型コンピュータシステムの第１コンピュータの動作方法であって、
（ａ）第１コンピュータに於いて、オブジェクトアプリケーションプログラムを格納する過程であって、各オブジェクトアプリケーションプログラムを以ってそのユーザが対応するデータタイプのオブジェクトに於けるデータを利用することを可能にする該過程と、
（ｂ）前記第１コンピュータに於いて、前記オブジェクトアプリケーションプログラムの第１オブジェクトアプリケーションプログラムを用いて、ユーザが対応する第１データタイプを有する第１オブジェクトに於けるデータを利用することを可能にする過程であって、前記第１

オブジェクトの前記データが、第２データタイプの第２オブジェクトを参照するようなリンクデータを含み、前記リンクデータが、前記第２オブジェクトが配置されているような第２コンピュータを識別する該過程と、
（ｃ）前記第１コンピュータに於いて、前記ユーザが前記リンクデータを選択することを可能にする過程と、
（ｄ）前記第１コンピュータと前記第２コンピュータとの間に第１通信リンクを確立し、前記第２オブジェクトに対応するデータタイプ情報を検索することを含めて前記第２コンピュータからの前記第２オブジェクトの検索を開始することによって、前記第１コンピュータが前記リンクデータのユーザ選択に応答する過程と、
（ｅ）前記第１コンピュータに於いて、前記第１コンピュータに格納された前記オブジェクトアプリケーションプログラムが前記第２データタイプに対応するオブジェクトアプリケーションプログラムを含むかどうか判定する過程と、
（ｆ）前記過程（ｅ）での判定結果が否定の場合、前記第２データタイプに対応するオブジェクトアプリケーションプログラムが前記第２コンピュータに格納されているかどうか判定する過程と、
（ｇ）前記過程（ｇ）での判定結果が肯定の場合、前記第２データタイプに対応する前記オブジェクトアプリケーションプログラムのコピーを前記第１コンピュータにロードする過程とを含むことを特徴とする方法。
【請求項４】　前記過程（ｆ）に於ける判定結果が否定の場合、前記第２データタイプに対応するオブジェクトアプリケーションプログラムが所定の他のコンピュータのセットの何れかの中に格納されているかどうか判定し、前記判定結果が肯定の場合、前記過程（ｇ）を実行する過程を更に含んでいることを特徴とする請求項３に記載の方法。
【請求項５】　コンピュータシステムと組み合わせて使用するためのコンピュータプログラムプロダクトであって、
該コンピュータプログラムプロダクトが、コンピュータ読取り可能記憶媒体と、そこに組み込まれたコンピュータプログラムメカニズムとを含み、
該コンピュータプログラムメカニズムが、
第２コンピュータに対応するコンピュータメモリに格納された第２オブジェクトへのリファレンスを前記ユーザが選択することを可能にするためのユーザインタフェース制御プログラムと、
第１コンピュータと前記第２コンピュータとの間に第１通信リンクを確立し、前記１つのオブジェクトに対応するデータタイプ情報を検索することを含めて前記第２コンピュータからの前記１つのオブジェクトの検索を開始することによって、オブジェクトリファレンスのユーザ選択に応答するためのコンピュータ間リンク制御プログラムとを含み、
前記ユーザインタフェース制御プログラムが、
前記第１コンピュータに対応するコンピュータメモリに、前記被検索データタイプに対応するオブジェクトアプリケーションプログラムが格納されて存在するかどうか判断し、前記判定結果が否定の場合、前記被検索データタイプに対応するオブジェクトアプリケーションプログラムを前記第２コンピュータに配置しようと試みるための検索命令を含み、
前記コンピュータ間リンク制御プログラムが、
前記検索命令が前記検索されたデータタイプに対応する前記オブジェクトアプリケーションプログラムを前記第２コンピュータに配置する場合、前記第１コンピュータに対応する前記コンピュータメモリに、前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムのコピーをロードするためのダウンロード命令を含むことを特徴とするコンピュータプログラムプロダクト。
【請求項６】　他のコンピュータと組み合わせて使用するためのコンピュータであって、
ビューワプログラムを格納するためのメモリであって、各ビューワプログラムを以って前記コンピュータのユーザが対応するデータタイプのビューオブジェクトを見ることを可能にする該メモリと、
前記他のコンピュータに格納されたオブジェクトを含めたオブジェクトへのリファレンスをユーザが選択することを可能にするユーザインタフェース制御プログラムと、
前記選択されたオブジェクトリファレンスに於ける情報に従って識別されるような前記他のコンピュータの１つへの第１通信リンクを確立し、前記参照されたオブジェクトに対応する検索データタイプ情報を含めた前記１つのコンピュータからの前記選択されたオブジェクトリファレンスによって参照される前記オブジェクトの検索を開始することによって、前記他のコンピュータに格納された前記オブジェクトの１つを参照するような前記オブジェクトリファレンスの１つのユーザ選択に応答するためのコンピュータ間リンク制御プログラムとを含み、
前記ユーザインタフェース制御プログラムが、
前記コンピュータに格納された前記ビューワプログラムが前記被検索データタイプに対応するビューワプログラムを含むかどうか判定し、前記判定結果が否定の場合、前記被検索データタイプに対応するビューワプログラムを前記１つのコンピュータに配置しようと試みるためのビューワ検索命令を含み、
前記ビューワ検索命令が前記被検索データタイプに対応する前記ビューワプログラムを前記１つのコンピュータに配置する場合、前記被検索データタイプに対応するビューワプログラムのコピーを前記コンピュータの前記メモリにロードするためのダウンロード命令を含むクラスローダを含むことを特徴とするコンピュータ。

【請求項７】　他のデバイスと組み合わせて使用するためのデバイスであって、
オブジェクトアプリケーションプログラムを格納するためのメモリであって、各オブジェクトアプリケーションプログラムを以って前記デバイスが対応するデータタイプのオブジェクトを利用することを可能にする該メモリと、
前記オブジェクトリファレンスに於ける情報に従って識別されるような前記１つのデバイスへの第１通信リンクを確立し、前記参照されたオブジェクトに対応するデータタイプ情報の検索を含めて、前記１つのコンピュータからの前記オブジェクトリファレンスによって参照される前記オブジェクトの検索を開始することによって、前記他のデバイスの１つに格納されたオブジェクトを参照するオブジェクトリファレンスに応答するためのデバイス間リンク制御プログラムと、前記デバイスの前記メモリに格納された前記オブジェクトアプリケーションプログラムが前記被検索データタイプに対応するオブジェクトアプリケーションプログラムを含むかどうか判断し、前記判定結果が否定の場合、前記被検索データタイプに対応するオブジェクトアプリケーションプログラムを前記１つのデバイスに配置しようと試みるための検索プログラムと、
前記プログラム検索が前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムを前記１つのコンピュータに配置する場合、前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムのコピーを前記デバイスの前記メモリにロードするためのダウンロード命令を含むクラスローダとを含むことを特徴とするデバイス。
【請求項８】　前記デバイス間リンク制御プログラムが、前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムの前記コピーをダウンロードするための前記１つのデバイスへの第２通信リンクを確立するための命令を含むことを特徴とする請求項７に記載のデバイス。
【請求項９】　データプロセッサを含むデバイスと組み合わせて使用するためのコンピュータプログラムプロダクトであって、
該コンピュータプログラムプロダクトが、コンピュータ読取り可能記憶媒体と、そこに組み込まれたコンピュータプログラムメカニズムとを含み、
該コンピュータプログラムメカニズムが、他のデバイスに格納されたオブジェクトを含めたオブジェクトへのリファレンスの選択を可能にする第１プログラムと、
前記選択されたオブジェクトリファレンスに於ける情報に従って識別されるような前記他のデバイスの１つへの第１通信リンクを確立し、前記被参照オブジェクトに対応するデータタイプ情報の検索を含めて、前記１つのデバイスからの前記選択されたオブジェクトリファレンスによって参照される前記オブジェクトの検索を開始することによって、前記他のデバイスに格納された前記オブジェクトの１つを参照する前記オブジェクトリファレンスの１つの選択に応答するためのデバイス間リンク制御プログラムと、
前記デバイスが前記被検索データタイプに対応するオブジェクトアプリケーションプログラムを格納するかどうか判断し、前記判定結果が否定の場合、前記被検索データタイプに対応するオブジェクトアプリケーションプログラムを前記１つのデバイスに配置しようと試みるための検索命令と、
前記ビューワ検索命令が前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムを前記１つのデバイスに配置する場合、前記被検索データタイプに対応する前記オブジェクトアプリケーションプログラムのコピーを前記デバイスにロードするためのダウンロード命令を含むクラスローダとを含むことを特徴とするコンピュータプログラムプロダクト。
【請求項１０】　前記コピーされたオブジェクトアプリケーションプログラムが、所定のオペランドスタック使用基準を含む所定のプログラムコレクトネス基準に適合するかどうか判定するためのベリフィケーション手順と、
前記ベリフィケーション手順によって前記コピーされたオブジェクトアプリケーションプログラムが前記所定のプログラムコレクトネス基準に適合するかどうか判定する場合、前記ユーザが前記被参照オブジェクトを見ることを可能にするべく、前記コピーされたオブジェクトアプリケーションプログラムの実行を可能にするためのプログラム許可命令とを更に含むことを特徴とする請求項９に記載のコンピュータプログラムプロダクト。
【手続補正３】
【補正対象書類名】明細書
【補正対象項目名】発明の詳細な説明
【補正方法】変更
【補正内容】
【発明の詳細な説明】
【００５５】図６乃至図１２、及び付録２を参照して、バイトコードベリファイヤプログラム２４０の動作について以下に詳細に説明する。付録２は、ベリファイヤプログラムを疑似コードで表したものである。付録２の疑似コードは、本質的には、汎用コンピュータ言語形式を用いたコンピュータ言語と同等である。この疑似コードは、本発明の説明のためにのみ作られたものであって、本分野の通常の知識を有するコンピュータプログラマであれば容易に理解されるように意図して書かれている。

【手続補正書】
【提出日】平成１５年１月１４日（２００３．１．１４）
【手続補正３】
【補正対象書類名】明細書
【補正対象項目名】００５５
【補正方法】変更
【補正内容】
【００５５】図６乃至図１２、及び付録２を参照して、バイトコードベリファイヤプログラム２４０の動作について以下に詳細に説明する。付録２は、ベリファイヤプログラムを疑似コードで表したものである。付録２の疑似コードは、本質的には、汎用コンピュータ言語形式を用いたコンピュータ言語と同等である。この疑似コードは、本発明の説明のためにのみ作られたものであって、本分野の通常の知識を有するコンピュータプログラマであれば容易に理解されるように意図して書かれている。